

Selling and Fulfillment Solutions Using WebSphere Commerce and IBM Sterling Order Management

**Sterling Commerce integration with
WebSphere Commerce**

**Business scenarios based on
latest feature pack**

**Sterling Commerce
business solution**



**Charlton Lee
Yumman Chan
Feras Dawisha
Sankar Kalla
Brenda Lam
Bhavin M. Majithia
Craig Oakley
Pankajkumar H. Patel**



International Technical Support Organization

**Selling and Fulfillment Solutions Using WebSphere
Commerce and IBM Sterling Order Management**

June 2011

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (June 2011)

This edition applies to WebSphere Commerce Version 7 Feature Pack 2, Sterling Order Management, Sterling Distributed Order Management.

© Copyright International Business Machines Corporation 2011. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	ix
Trademarks	x
 Preface	 xi
The team who wrote this book	xi
Now you can become a published author, too!	xiii
Comments welcome	xiv
Stay connected to IBM Redbooks	xiv
 Chapter 1. Introduction and product overview	 1
1.1 Sterling Order Management	2
1.1.1 Improved supply chain efficiencies and revenue growth.	2
1.1.2 What Sterling Order Management can do to address the unique concerns of today's advanced order and fulfillment processes	2
1.1.3 How Sterling Order Management can help you achieve cross-channel excellence	4
1.2 Sterling Distributed Order Management	4
1.2.1 Sourcing and scheduling orders intelligently (and globally)	4
1.2.2 Managing and coordinating customized fulfillment processes	5
1.2.3 Flawlessly executing beyond the four walls	6
1.2.4 Providing a single source of order information	6
1.2.5 IBM Sterling Distributed Order Management module capabilities. ...	6
1.3 IBM Sterling Global Inventory Visibility	7
1.3.1 Global business advantages	8
1.3.2 Advanced inventory control system	8
1.3.3 Monitoring inventory	8
1.4 WebSphere Commerce	9
1.4.1 Support for multiple business models	9
1.4.2 Allows creation of custom sites for specific customers and downstream partners	10
1.4.3 Deploying a single, consolidated platform for e-commerce	11
1.4.4 What analysts say about WebSphere Commerce	11
1.4.5 Cross-channel order management	12
1.4.6 Social Commerce	13
1.4.7 Commerce extended sites	14
1.4.8 Robust B2B and B2C	15
1.5 WebSphere Commerce Distributed Order Management	16
1.6 References	18

Chapter 2. Sterling-Commerce solution overview	19
2.1 Solution overview	20
2.2 WebSphere Commerce architecture	20
2.2.1 Functional architecture	21
2.2.2 Multi-channel presentation layers	22
2.2.3 Application architecture	23
2.2.4 WebSphere Commerce framework overview	25
2.3 WebSphere Commerce DOM cross-channel integration	26
2.3.1 Commerce DOM integration detailed usage scenarios	28
2.3.2 Reliability and performance strategy	30
2.3.3 Stock location	30
2.3.4 Transfer order service requests	30
2.4 WebSphere Commerce integration architecture	31
2.4.1 Built-in DOM store and inventory integration	31
2.4.2 WebSphere Commerce DOM integration specifics	33
2.4.3 Subsystem end-to-end flow	35
2.5 Enterprise Service Bus (ESB)	36
2.5.1 IBM SOA Reference Architecture	36
2.5.2 SSFS and WebSphere Commerce integration via ESB	38
2.5.3 WebSphere Enterprise Service Bus	39
2.5.4 WebSphere Integration Developer	41
2.5.5 WebSphere Message Broker	41
2.5.6 DataPower	43
2.6 Design considerations	44
Chapter 3. Business scenarios	47
3.1 Business scenarios overview	48
3.2 SSFS WebSphere Commerce scenarios	48
3.2.1 Order browse	49
3.2.2 Order capture	49
3.2.3 Order status	50
3.2.4 Order cancel	51
3.2.5 Order modification	52
3.2.6 Returns processing	54
3.3 Available features in WC	55
3.3.1 Store locator	55
3.3.2 Stock locator	55
3.3.3 Buy-online-pick-up-in-store	55
3.3.4 Buy-online-ship-to-store	56
3.3.5 Reserve online pay in-store	56
3.3.6 DOM integration	56
3.4 Scenarios covered OOB	56

Chapter 4. Business scenario: Catalog Browse	57
4.1 Scenario introduction	58
4.2 Prerequisites	58
4.3 Catalog browse flow	59
4.3.1 System interaction diagram	60
4.3.2 Flow diagram	61
4.3.3 Actions and subsystems in the scenario	62
4.3.4 Execution flow	62
4.3.5 Alternative buy-online-and-pickup-in-store (BOPIS) flow	67
4.3.6 Exception flow verification	72
Chapter 5. Business scenario: Order capture	79
5.1 Scenario introduction	80
5.2 Prerequisites	82
5.3 Adding item to cart	82
5.3.1 Flow diagram of adding item to a cart	83
5.3.2 Actions and subsystems in the scenario	84
5.3.3 Scenario overview and system impact results	84
5.3.4 Execution flow	85
5.3.5 Exception flows while entering item to a cart	90
5.4 Checkout flow	91
5.4.1 System integration diagram	91
5.4.2 Flow diagram	92
5.4.3 Actions and subsystem in the scenario	93
5.4.4 Scenario overview and system impact results	93
5.4.5 Execution flow	95
5.4.6 Flow verification	101
5.5 Buy-online-pickup-in-store (BOPIS) scenario	105
Chapter 6. Business Scenario: Order status	113
6.1 Scenario introduction	114
6.2 Prerequisites	114
6.2.1 Enabling the order status in the management center	115
6.2.2 User should be a registered user	121
6.3 Order status flow	121
6.3.1 System interaction diagram	122
6.3.2 Actions and subsystems in the scenario	122
6.3.3 Execution flow	123
6.4 Sterling order fulfillment	127
Chapter 7. Installation and configuration	129
7.1 Introduction	130
7.1.1 Integration overview	130
7.1.2 Installation, configuration, and deployment	132

7.2	WebSphere Commerce installation and configuration	132
7.3	WESB mediation module installation and configuration	133
7.3.1	Importing the mediation module into WID	134
7.3.2	Making necessary changes to the mediation module	135
7.4	Installing, configuring, and deploying SSFS	140
7.4.1	Enabling inbound API calls over JMS	140
7.4.2	Enabling outbound API calls over JMS	145
7.5	Configuring SSFS for integration with WebSphere Commerce	157
7.5.1	Configuring participants	158
7.5.2	Catalog management	167
7.5.3	Global Inventory Visibility application	169
7.5.4	Distributing Order Management	174
7.6	Integration flow data mapping	182
Chapter 8.	Integration implementation	195
8.1	WebSphere Commerce DOM inventory cache management	196
8.1.1	Inventory availability cache	196
8.1.2	Caching examples	198
8.1.3	Cache timing	198
8.1.4	Inventory availability cache tables	199
8.1.5	Inventory cache schemas	199
8.1.6	INVCNF configuration and test results	203
8.1.7	DOM interfaces	207
8.2	WebSphere Enterprise Service Bus overview	207
8.2.1	WebSphere Enterprise Service Bus key terms	208
8.2.2	Mediations, service consumers, and service providers	208
8.2.3	Mediation modules	210
8.2.4	Mediation flow components	211
8.2.5	Mediation flows	211
8.2.6	Mediation primitives	213
8.3	WCToSSFSMediationModule: Processing of order capture	214
8.3.1	WCToSSFSMediationModule mediation module	215
8.3.2	WCToSSFSMediationModule mediation flow	219
8.3.3	WCToSSFSMediationModule mediation flow request	221
8.3.4	WCToSSFSMediationModule mediation flow request primitives	221
8.3.5	WCToSSFSMediationModule mediation flow response	223
8.4	WebSphere Commerce DOM implementation	223
8.4.1	WebSphere Commerce commands behavior with business rules	223
8.4.2	Default WebSphere Commerce commands behavior	231
8.4.3	Security and authentication	231
8.4.4	Database persistence	231
Appendix A.	Supporting content	233

Glossary 241

Related publications 243

IBM Redbooks 243

Online resources 243

Help from IBM 245

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:


This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

DataPower®
DB2®
IBM®

Lotus®
Redbooks®
Redbooks (logo) ®

Tivoli®
WebSphere®

The following terms are trademarks of other companies:

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

This IBM® Redbooks® publication brings together subject matter experts with experience using the leading IBM customer interaction platform for cross-channel and online commerce, IBM WebSphere® Commerce, with the powerful IBM Sterling Order Management, which coordinates order fulfillment from all channels and across the extended enterprise. An integrated solution was built in the lab that illustrates how these products can be integrated to benefit IBM customers.

This publication focuses on the integration of the IBM high-volume commerce solution designed to address enterprise commerce needs by delivering a rich, robust multi-channel customer experience, with Sterling Order Management, designed to enable supplier collaboration with management and order fulfillment process optimization. By integrating WebSphere Commerce and Sterling Order Management with out-of-the-box components, we prove that customers are provided an end-to-end solution to address a complete opportunity for a fulfillment life cycle that is cost effective and easy to implement.

This publication targets a technical audience for the documentation of the integration approach by explaining the solution architecture and the implementation details. However, this publication also contains introductory chapters that contain executive summary material and provides well-documented scenarios with use cases for business analysts whose domain would be these systems.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.

Charlton Lee is a Senior Solution Architect for Software Group Smarter Commerce Center of Excellence, and previously was SOA Sales Manager for IBM ASEAN/AP. An Integration Specialist with WebSphere Commerce, he has 25 years of IT consulting experience delivering large system solutions for worldwide enterprises. Being well versed with WebSphere infrastructure products (DataPower/WESB) and other third-party technologies, he has co-authored the IBM Redbooks publication *DataPower Architectural Design Patterns: Integrating and Securing Services Across Domains*, SG24-7620.

Yumman Chan is the Chief Architect for the Industry Solution Services Team. He has more than 20 years of IT experience and is one of the founders of WebSphere Commerce. His interest is to map customer requirements into solution architecture and provide actual customer requirement feedback to product development in order to drive down the total cost of solution implementation.

Feras Dawisha is a Practice Manager for the Industry Solution Services Team. He is a Certified IT Architect with over 20 years of experience in the IT industry, 12 of which were with WebSphere Commerce. His speciality is in Commerce Solution Methodology and Enterprise Architecture.

Sankar Kalla is a WebSphere Commerce Expert working for SAPIENT Ecommerce labs in the USA. He has five and half years of experience in the e-Commerce field. His areas of expertise include Java™, J2EE, AJAX, SOA, REST, and external system integration with WebSphere Commerce. He has written extensively on WebSphere Commerce technology. He has a master's degree from BITS Pilani India.

Brenda Lam is a Senior Solution Architect for the Industry Solution Services team. She has over 20 years of IT experience, 13 of which working on commerce solutions. She was previously the Chief Architect for the ibm.com Commerce Team and currently specializes in telecommunication solutions.

Bhavin M. Majithia is a Software Specialist with IBM India Software Labs at Bangalore. He has six years of experience in WebSphere and JEE technologies. Bhavin has a master's degree in Computer Applications from T.N.Rao College (Saurashtra University), Rajkot, and is certified in IBM WebSphere Commerce. He has written extensively on WebSphere Commerce solutions.

Craig Oakley is a IT Specialist Professional Certification (L2) in Application Integration and Middleware. He is the WebSphere Commerce Services Lead for Australia/New Zealand. He has a developer background. Prior to working at IBM, he was the Lead Consulting Architect for Telstra Shop Online v2 (currently live) the Lead Administrator for Telstra Shop Online (replaced by Telstra Shop Online v2), and a Subject Matter Expert (SME) for WebSphere Commerce Performance and Security. Craig served as a guest speaker about WebSphere Commerce at WSTC 2007 and 2008. Craig was the Lead Consulting Architect on WebSphere Commerce projects for Flight Center (travel), The Good Guys (electrical), and Cameras Direct (photography). He was also a Consulting Architect for WebSphere Commerce projects for Coles (supermarket) and KMart.

Pankajkumar H. Patel is a Staff Software Engineer working with the WebSphere Portal and Web Content Management Support Team at the IBM Pune, India, facility. He has a bachelor's degree in Information Technology from Bhavnagar University and is currently pursuing his MBA in International Business. He is also

an IBM WebSphere Portal, Web Content Management, and Sun Certified Solution Developer. He has extensive experience in Portal and related technologies and has worked as a J2EE Developer utilizing IBM rational tools and technologies before joining IBM in 2005. You can reach him at panpatel@in.ibm.com.

Thanks to the following people for their contributions to this project:

Rufus P. Credle Jr, Tamikia Barrow, and Linda Robinson
International Technical Support Organization, Raleigh Center

Thomas Obremski, Ph.D., Sterling Commerce Portfolio Integration Manager,
IBM Software Group, IBM Cambridge

Rod Martinez, Senior Product Marketing Manager, Order Management, Sterling
Commerce
IBM Denver

Adam Orentlicher, IBM-Sterling Commerce Development Integration Lead
IBM Research Triangle Park

Danai Tengtrakool, IBM Software Group, Industry Solutions Development
Software Architect
IBM Lowell

Venkateswarlu Avvaru, IBM Software Group, Industry Solutions Development
IBM Lowell

Laura Apostoloiu, Manager Performance and SVT, WebSphere Commerce
IBM Toronto

Peter Crocker, Senior Manager, WebSphere Commerce Development
IBM Toronto

Sonny Sia, Software Group Services Manager
IBM Toronto

Alex Shum, WebSphere Commerce Development
IBM Toronto

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using

leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

ibm.com/redbooks

- ▶ Send your comments in an e-mail to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- ▶ Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- ▶ Look for us on LinkedIn:

<http://www.linkedin.com/groups?home=&gid=2130806>

- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- ▶ Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>



Introduction and product overview

In this chapter, we introduce and discuss IBM products that strengthen the integration of the IBM high-volume commerce solution. The IBM high-volume commerce solution is designed to address enterprise commerce needs by delivering a rich, robust multi-channel customer experience, with IBM Sterling Order Management suite, designed to enable supplier collaboration with management and order fulfillment process optimization. By integrating IBM WebSphere Commerce and IBM Sterling Order Management, clients are provided with an end-to-end solution to address the complete opportunity to fulfillment life cycle.

1.1 Sterling Order Management

Amid increased competitive pressure and growing customer demand, fulfilling orders across an extended supply chain has become increasingly complex. Sterling Order Management can help you manage these complexities, allowing your organization to improve supply chain efficiencies and business responsiveness by cost-effectively orchestrating global product and service fulfillment across the extended enterprise.

1.1.1 Improved supply chain efficiencies and revenue growth

Sterling Order Management provides robust multi-channel order management functionality that can:

- ▶ Intelligently broker orders across many disparate systems.
- ▶ Provide a global view of all inventory across the supply chain.
- ▶ Help you make changes to business processes on the fly.

Through the use of an intelligent sourcing engine, a central order repository, and the aggregation of global inventory, Sterling Order Management can help you grow revenue and become best-in-class by cost-effectively orchestrating global order and service fulfillment across the extended enterprise.

Why is complex order management needed now more than ever? Customers are demanding a unified shopping experience, creating a much more complex supply chain. Delivering innovative services such as *buy online* and *in-store pickup*, has forced companies to incorporate new processes that support cross-channel visibility and customer order fulfillment. Also, competitive pressures continue to force your organization to become more efficient in order fulfillment and services across your extended supply chain.

As orders are fulfilled across multiple internal entities and external partners, it becomes difficult to efficiently manage all the processes needed to provide a uniform customer experience. Many organizations rely on inefficient manual processes to complete transactions that cross channels. Simultaneously, a lack of inventory visibility across all locations can result in exceptionally high stock-outs and inefficient inventory utilization.

1.1.2 What Sterling Order Management can do to address the unique concerns of today's advanced order and fulfillment processes

Sterling Order Management addresses the multi-channel complexities of modern order fulfillment across the extended enterprise. An intelligent sourcing engine

looks across all locations, including external partners, to determine the best location to fulfill each line on the order, based on a wide set of parameters that your organization chooses. Sterling Order Management identifies the applicable fulfillment process for each order and seamlessly splits or consolidates order lines and sequence activities. It brokers documents and requests to the appropriate internal or external fulfillment participants and incorporates user-defined events to effectively track fulfillment activity based on the unique conditions of each order line.

Associated services can also be scheduled at the same time as the order, increasing the amount of revenue from each sale. If your company provides delivery services, on-site setup, or after-sales service, it can be scheduled at the same time at which the order is being placed. With delivery and service scheduling, a range of value-added services (provided internally or from a third party) can be sold to the customer and scheduled along with the products. The solution tracks crew capacity for taking appointments and ensures that the service technician is at the correct location when the item is delivered.

Sterling Order Management also enables you to efficiently manage the returns process. Pre-defined business process flows ensure that returned products are consistently handled in the proper manner and that no items are lost or forgotten in the process. This ensures that your organization efficiently utilizes all inventory, thus reducing your overall inventory costs.

Sterling Order Management gives your company a single comprehensive view of all inventory information by aggregating inventory from all locations and providing a view of what is available internally as well as at all partner locations, what is being supplied, what is in transit, and what the current demand is. This extensive visibility ensures that you are giving your customers an accurate promise date for all their orders, and your inventory is being utilized in the most efficient way (Figure 1-1).

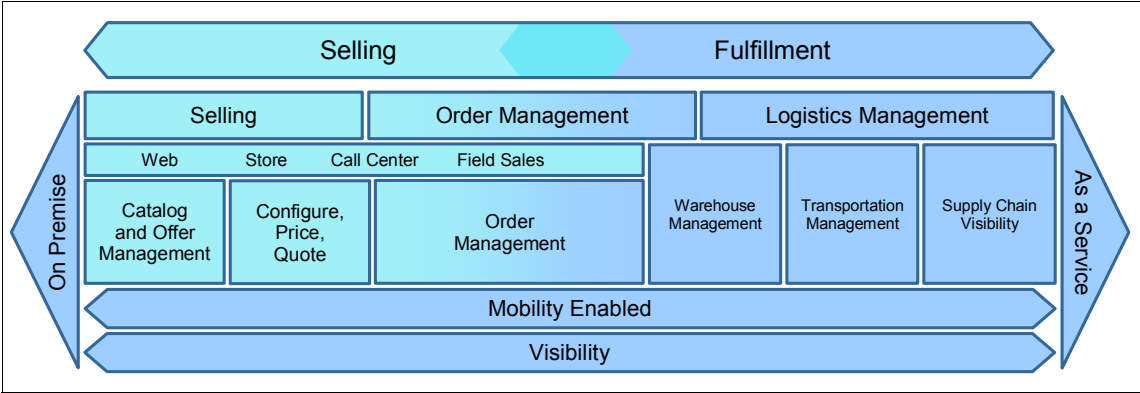


Figure 1-1 Sterling Order Management

The Sterling Order Management solution incorporates best practices to allow organizations to quickly improve their fulfillment processes with out-of-the-box line-level order fulfillment, whether it is for the delivery of goods or the coordination of on-site services. Also, using a graphical tool within Sterling Order Management's business process modeler, your organization can quickly and efficiently add or change participants in the order process to meet changing business needs without requiring changes in the rest of your fulfillment network.

1.1.3 How Sterling Order Management can help you achieve cross-channel excellence

The robust functionality of Sterling Order Management can be utilized across multiple selling channels, which include contact (call) center, store, web, and field sales. Each of these channels uses Sterling Order Management to place or modify orders, determine order status, check inventory availability across all locations, and manage the returns process. Utilizing these channels will allow your company to provide enhanced cross-channel services to your customers and partners, allowing them to begin any type of transaction in any channel, and complete it in another channel. For example, a customer can begin a shopping transaction (browse an online catalog and initiate a shopping cart) on a web storefront and complete the transaction with the help of an employee in the store (or a call center representative), or they can order items over the phone and return them through the web storefront. Sterling Order Management helps you manage the complexities of global order fulfillment, allowing your company to achieve cross-channel excellence.

1.2 Sterling Distributed Order Management

Sterling Distributed Order Management brings order to supply chain and fulfillment complexity. It enables your company to execute and coordinate order fulfillment processes across your extended supply chain network. It provides flexible, process-based management of orders from multiple channels and enables customized fulfillment based on user-defined business requirements. Sterling Distributed Order Management delivers the required visibility and event management across all fulfillment activity, allowing you to respond quickly to unexpected problems and meet customer expectations.

1.2.1 Sourcing and scheduling orders intelligently (and globally)

Most legacy systems were designed for order management applications to be linked to a discrete number of specific plants or warehouses. However, this limits

inventory visibility and fails to account for deliveries and associated services that are increasingly part of the customer order and fulfillment process. Without visibility to all internal and external inventory locations and consideration for delivery and service requirements, it is impossible to provide an accurate promise to the customer or schedule orders to alternative fulfillment nodes — ones that might be better suited for a particular customer situation or to lower costs.

Sterling Distributed Order Management combines multi-channel order aggregation with global visibility to inventory, delivery, and service availability, enabling the complete order promise (available-to-promise, available-to-deliver, and available-to-service) and providing the ability to “order from anywhere and fulfill from anywhere.” With optimized, rules-based order promising and scheduling, inventory and resources are appropriately allocated from any internal or external source to meet the conditions of the order and the requirements of your business. It is the first step to optimizing fulfillment processes based on rules tied to your company’s performance objectives.

1.2.2 Managing and coordinating customized fulfillment processes

Order fulfillment is not the simple, highly repetitive process of the past. Customer demands for customization have quickly extended to fulfillment. For example, a manufacturer of mobile phones must label, provision, and package the same phone differently for each wireless carrier. In other cases, customers look to the selling enterprise to coordinate associated services, such as testing, delivery, and installation. Even more complex is the execution of customized offerings, such as unique products or dynamic bills-of-materials that require successful coordination of configure-to-order or build-to-order processes. Companies that efficiently execute these processes will make the complexity transparent to the customer, dramatically improving their customer relationships and differentiating their value proposition.

With event management and configurable business rules and fulfillment processes, Sterling Distributed Order Management enables customized, line-level order fulfillment. It de-composes orders into unique “units of work” for fulfillment, whether that be inventory movement or coordination of delivery and on-site services. Given defined conditions, Sterling Distributed Order Management identifies the applicable fulfillment process for each unit of work, and seamlessly splits and consolidates order lines and sequence activities. It brokers documents and requests to the appropriate internal or external systems and participants and incorporates user-defined events to effectively track fulfillment activity based on the unique conditions of each order line.

1.2.3 Flawlessly executing beyond the four walls

Fulfillment is no longer contained within a single enterprise. For most companies, the fulfillment process involves multiple parties performing various activities throughout the order life cycle. When an order leaves a business's four walls, businesses lose visibility and control. However, the customer's expectations still reside with the selling company. Sterling Distributed Order Management delivers flexible order process modeling and establishes role-based relationships among multiple supply chain participants. It coordinates accurate fulfillment based on these relationships and controls critical information flows such as orders, modifications, status updates, and exceptions.

1.2.4 Providing a single source of order information

Customer expectations can only be managed with accurate and timely information. Due to the increase in order capture channels powered by different systems, accurate order information is often unavailable when needed. Because companies maintain multiple databases of order information, they are forced to manage by individual channel, rather than across channels. Sterling Distributed Order Management aggregates orders from multiple order capture channels and provides a single source of information across these channels. It enables your company to present a single face to the customer by allowing information about any order, from any channel or division, to be made available when and where a customer needs it. It also simplifies administration and maintenance of customer orders, allowing a single record to be accessed, modified, or cancelled through simplified integration between any order capture system and the Sterling Distributed Order Management application. All information and activity related to that order is contained in a single repository, presenting a single version of the record.

1.2.5 IBM Sterling Distributed Order Management module capabilities

In this section, we discuss the capabilities of Sterling Distributed Order Management, which are:

- ▶ Distributed order management
 - Aggregates, manages, and monitors orders from all channels
 - Intelligent sourcing engine coordinates fulfillment across the extended enterprise
 - Provides a single order repository to modify, cancel, track, and monitor the order life cycle in real-time

- ▶ Delivery and service scheduling
 - Dynamically schedules product deliveries and associated services at the time of sale based on the order conditions and resource availability
 - Monitors service and delivery execution based on agreed-upon service parameters
- ▶ Inventory synchronization
 - Provides visibility of supply and demand across all internal and external locations
 - Configures inventory categories to meet specific business requirements
 - Provides real-time, global available-to-promise checks
- ▶ Inventory visibility
 - Get personalized and real-time access to inventory information.
 - Provide users with role-specific views of inventory.
 - Leverage advanced inventory search capabilities.
- ▶ Reverse logistics
 - Links multiple return/repair requests to original sales orders to enable repair life-cycle tracking
 - Tracks reverse inventory back to the appropriate location, including partner locations, based on flexible business rules
- ▶ Business process definition framework
 - Graphically configures unique business processes
 - Quickly defines relationships and roles of participating organizations
 - Connects to internal and external supply participants and systems

1.3 IBM Sterling Global Inventory Visibility

Sterling Global Inventory Visibility gives your company a single comprehensive view of all inventory information by aggregating inventory from internal and external locations and providing a view of what is available within your organization and all partner locations, what is being supplied, what is in transit, and the current demand. This extensive visibility ensures that you are giving your customers an accurate promise date for all of their orders and that your inventory is being utilized in the most efficient way.

Sterling Global Inventory Visibility can also be used across multiple channels. Inventory availability information can be accessed from a call center, store, or the

web, and a field sales agent can access it remotely, ensuring that all supply chain participants are utilizing the most up-to-date inventory information.

1.3.1 Global business advantages

As part of the IBM Sterling Selling and Fulfillment Suite solution, Sterling Global Inventory Visibility can be used across multiple products, depending on the type of business problem that it is intended to solve. For example:

- ▶ IBM Sterling Order Management provides available-to-promise capabilities for customer orders, based not only on what inventory is on hand, but also what has been ordered and is in transit.
- ▶ IBM Sterling Supply Chain Visibility provides on-hand and projected inventory levels based on what has been ordered and in transit, so as to provide decision support for prioritizing, expediting, or diverting inbound supply or outbound orders.

1.3.2 Advanced inventory control system

Typical inventory control systems are managed locally and are focused on tactical processes such as day-to-day inventory management and cycle counting. Such systems are not designed to provide strategic decision support capabilities that require a global view of inventory information across different locations and systems. Current systems are also limited in their ability to supply access control for inventory systems and users beyond the four walls of the location that they serve.

Sterling Global Inventory Visibility not only handles the tactical process, but the visibility capabilities include a forward-looking, time-phased inventory view accessible through a comprehensive inventory console. Use of this inventory visibility solution presents you with a central repository of inventory information, which can collect real-time data from other systems, then serve as a single source that can be accessed by multiple users.

1.3.3 Monitoring inventory

Inventory in multiple locations can be monitored to ensure the most efficient utilization and the prevention of obsolete inventory. On-hand inventory monitors can be time-triggered to ensure that items do not spoil or become obsolete. Event-based monitors can also determine availability across both internal and externally owned inventory.

1.4 WebSphere Commerce

IBM WebSphere Commerce is a key component of the IBM e-commerce solution. WebSphere Commerce provides a single, integrated platform to support the many ways that a company does business and to meet the challenges unique to cross-channel e-commerce. WebSphere Commerce provides the following differentiated capabilities to support the business goals of e-commerce and to enable the best practices of successful e-commerce business models.

1.4.1 Support for multiple business models

WebSphere Commerce can support the widest range of out-of-the-box business models, and can do so on a single installation. Companies need solutions that accommodate all of the ways that they do business, without having to invest in multiple third-party software or taking on expensive customization or re-engineering tasks. WebSphere Commerce's architecture and prepackaged, customizable configurations enable it to provide the broadest range of out-of-the-box B2C, B2B, and multi-channel business models, and those business models can be configured, customized, and extended to meet your needs and characteristics. New business models can even be created by combining elements from those that were preconfigured.

The WebSphere Commerce architecture allows multiple sites to share infrastructure and resources, allowing them to achieve economies of scale, yet maintain tight security between them. And the robust underlying WebSphere platform helps ensure that the installation can handle the workload and scale appropriately for cross-channel performance and availability.

WebSphere Commerce provides a common interface for multiple customer touchpoints. It has a Business Context Engine component that presents a single face to the business, no matter what part of the business process the user interacts with and no matter which channel or touchpoint they use. Whether they shop online, in the store, via kiosk, or telephone, eCommerce customers get a consistent, highly personalized experience that meets or exceeds their demands for maximum choice and convenience.

By deploying a single, consolidated platform for e-commerce, you can configure this patented technology to reflect the characteristics that define and differentiate your business:

- ▶ Market segmentation
- ▶ Business policies
- ▶ Supported roles and workflows

- ▶ Inter-company agreements
- ▶ Regional locales
- ▶ And much more

Every channel or touchpoint that is powered by WebSphere Commerce can leverage common business rules, product and customer data, user interfaces and marketing and merchandising tactics. This means significant cost savings, reduction in development time, and an improved cross-channel synergy and customer experience.

1.4.2 Allows creation of custom sites for specific customers and downstream partners

For an unparalleled degree of individualized service, you can create branded, custom stores for specific customers with the WebSphere Commerce Extended Sites capability. Each of these sites can appear unique to the customers that access it, and each site can implement business rules and policies unique to the customer relationship (special pricing and product entitlement, for example). A site can even be integrated with in-house procurement systems. The unique sites co-exist on the same infrastructure, sharing as much data and business logic as possible, so that the deployment and management of the operation is quick, easy, and cost-effective.

With the same solution, you can deploy a storefront service for downstream channel partners (such as resellers, distributors, agents, and dealers). These partners can more quickly and easily create and customize their own commerce sites, populating your catalog from a master and adding their own items. The extended sites capability lets you host a virtually unlimited number of distinct websites. Partners get expanded reach and improved customer satisfaction and loyalty, and you get brand integrity, order, and inventory visibility and insight into customers.

WebSphere Commerce was designed to provide a dynamic global infrastructure to help accommodate unique customer, business, language, and legal requirements for every region or country in which a business operates. It supports multiple currencies, local tariffs, shipping rules, and more. The product is shipped in 10 languages and is enabled to support most single-byte and double-byte languages. Companies can use WebSphere Commerce to conduct business around the world more efficiently and provide better customer service and support—all from a single e-commerce site.

1.4.3 Deploying a single, consolidated platform for e-commerce

WebSphere Commerce Starter Stores deliver preconfigured sets of business processes that you can adapt to specific requirements as your business needs dictate. This lets you perform minimal customization to a new website, helping you deploy sites more quickly. Once the platform is in place, you can deploy fully configured, production-ready websites in days instead of months, and wizards, tools, and role-based access control equip your employees to brand, configure, customize, and operate e-commerce sites with ease.

1.4.4 What analysts say about WebSphere Commerce

AMR Research ranked WebSphere Commerce ahead of its competition by noting its key strengths:

- ▶ Exceptional multi-site capabilities for deploying multi-brands and microsites via extended site functionality
- ▶ Robust international capabilities
- ▶ Highly functional B2B capabilities along with robust B2C e-commerce functionality
- ▶ Advancing the Web 2.0 model
- ▶ Offering RIA-based single-page checkout and other widgets out of the box
- ▶ Mature and stable platform

Gartner, Inc. evaluated the top vendors in e-commerce and positioned IBM as the leader in e-commerce. Once again, Gartner named WebSphere Commerce in the Leaders Quadrant for E-Commerce Magic Quadrant. Gartner defines leaders as "technology providers that demonstrate the greatest degree of support for B2B and B2C Internet sales."

Highlights from Gartner Magic Quadrant for E-Commerce:

- ▶ IBM provides a wide array of core e-commerce capabilities out of the box, as well as extended capabilities and aspects of Web 2.0 single-page check-out that is, pre-integrated into the Web 2.0 store.
- ▶ IBM's significant customer base, and the large array of industries and geographies where the product is being used, are recognized as strengths.
- ▶ IBM's large network of implementation partners is recognized as a strength against other E-Commerce vendors.

1.4.5 Cross-channel order management

More companies cite as a top concern the lack of integration of inventory and order management systems. Most companies have multiple order and inventory systems in place. A complex legacy environment such as this prevents easy, seamless access to order and inventory information. Yet customers want to place, view, or modify orders and returns in any channel that they choose, and they want an instant view of availability before placing an order or visiting the store. Non-integrated order and inventory systems hinder companies from responding to customer expectations.

The subsystem has been extended and improved to provide enhanced support for cross-channel business processes and efficiently serve new constituents such as contact center representatives, gift registrants, and distribution channel partners. Additional integration interfaces have been added to facilitate communication with external systems (for example, POS systems, kiosks, Enterprise Resource Planning, and fulfillment systems), and a new plug-in based payment processing capability has been added.

WebSphere Commerce and Sterling Order Management

The WebSphere Commerce (Version 7 Feature Pack 2) and Sterling Order Management Integration kit is a collaborative development project between IBM and Sterling Commerce to provide two-way integration between IBM WebSphere Commerce and the Sterling Order Management solution. The combined solution enables a unified customer experience by spanning application and enterprise boundaries to combine fragmented financial, inventory, and logistics processes into a single, central view across all business units and channels. This delivers a high-level of customer satisfaction by combining:

- ▶ WebSphere Commerce capture of orders across various channels, including the web channel (customer or customer service representative) and the call center channel.
- ▶ Orchestration of order processing by Sterling Order Management to choose the most cost-effective fulfillment channels. The selection process includes identifying sources of supply (such as internal warehouses or third-party suppliers), scheduling delivery and associated services, and executing fulfillment.

When you deploy WebSphere Commerce and Sterling Order Management together, you get capabilities that lead the industry in order capture and order fulfillment without the pain of integrating them. The Integration Kit accelerates the return on investment by getting you into production faster.

This solution uses services-oriented architecture (SOA) to provide industry-leading technology for order capture and complex order-management problems. WebSphere Commerce and the Sterling Order Management solution that reduce time to market as well as total cost of ownership because of the flexibility that SOA provides. By automating the orchestration of order fulfillment across multiple channels, you can offer your customers greater flexibility with less cost to you.

1.4.6 Social Commerce

Social commerce has taken word-of-mouth where it never really existed before, the online shopping world. Customers now are looking for ways to leverage each other's expertise, understand what they are purchasing, and make more informed and accurate purchase decisions. Retailers need to understand their customers and what they expect out of the shopping experience to develop a successful social commerce strategy.

Social Commerce in WebSphere Commerce Version 7 enables the creation of user-generated content and tracking the creation of social content for marketing and community building purposes. The types of user-generated content that can be created out-of-the-box include blogs, social profiles, ratings and reviews, and photo galleries. Through platform providers such as Pluck, additional modules such as forums, videos, comments, and other applications can also be deployed.

There are two categories of social software that can be leveraged from WebSphere Commerce:

- ▶ On-premise licensed social software

With this type of integration, retailers purchase, install, and manage the on-premise social software that will be used by WebSphere Commerce to store and retrieve social content. With Version 7 of WebSphere Commerce, there is pre-built integration with IBM Lotus® Connections 2.5 for this purpose. The retailer will be responsible for provisioning the hardware and software for Lotus Connections. The retailer also is responsible for managing the infrastructure and configuring the software.

- ▶ Social Software as a Service (SaaS)

With this type of integration, the retailer is expected to sign a Service Level Agreement (SLA) with a vendor that hosts and manages social content on behalf of the retailer. There is support in WebSphere Commerce Version 7.0 for two popular vendors:

- Pluck
- Bazaar Voice

The integration with Pluck enables the use of blogs, photo galleries, and social profiles. Pluck also enables discovery, comments, video, groups, forums, ratings and reviews, and custom web, mobile, and desktop applications running on its social application server. Integration with Bazaar Voice enables the use of ratings and reviews with store assets. Additional SaaS vendors can be added by extending the Social Commerce solution to a specific SaaS vendor's content API.

WebSphere Commerce Version 7.0 introduces or extends interactive Web 2.0 widgets that render content without page transitions, creating a versatile, customer-friendly store. These widgets can use the same themes and styles as the rest of the store pages without any coordination with the social software vendors. The architecture also uses a Representational State Transfer (REST) style that allows the retailer to leverage a common social API for extending the solution. It also allows the use of standard caching technologies. The solution enables rendering stylized Search Engine Optimized (SEO) representation of dynamic social content to improve the page ranking of the store pages associated with the social content. Being out-of-the-box functionality, it is possible to leverage the underlying IBM WebSphere sMash technology for Social Commerce to integrate with other third-party vendors or in-house software, without changing any of the store page integration.

1.4.7 Commerce extended sites

WebSphere Commerce extended sites capability enables you to create multiple, unique sites to serve different brands, regions, or targeted groups of customers. Each of these sites can have a unique look and feel and can implement business rules and policies unique to the customer relationship (special pricing and product entitlement, for example). Yet the unique site coexist on the same infrastructure, sharing as much data and business logic as possible to ease operation and management. You can also use the extended sites capability to enable downstream channel partners to create and manage their own e-commerce sites with easy-to-use, web-based tools. You can control site presentation as well as catalog content, or you can allow partners to customize the sites and catalogs.

Extended site features include:

- ▶ Supports the widest range of out-of-the-box business models, even allowing different business models on a single installation.
- ▶ Allows creation and hosting of a virtually unlimited number of custom sites for specific customers and downstream partners.

- ▶ Displays each site uniquely to the customers that access it, and each site can implement business rules and policies unique to the customer relationship (special pricing and product entitlement, for example).
- ▶ Leverages common business rules, product and customer data, user interfaces, and marketing and merchandising tactics to every channel or touch point that is powered by WebSphere Commerce extended sites.
- ▶ Deploys a storefront service for downstream channel partners (such as resellers, distributors, agents, and dealers). These partners can quickly and easily create and customize their own e-commerce sites, populating the catalog from a master and adding their own items.
- ▶ Deploys regional sites to accommodate unique customer, business, language, and legal requirements for every region or country in which a business operates.

Benefits of extended sites:

- ▶ Significant cost savings because sites use the same infrastructure, sharing as much data and business logic as possible.
- ▶ Vast reduction in development time for new sites.
- ▶ Improved cross-site and cross-channel synergy and customer experience.
- ▶ An unparalleled degree of individualized service.
- ▶ Partners using the storefront creation and hosting capabilities get expanded reach and improved customer satisfaction and loyalty, and you get brand integrity, order, and inventory visibility and insight into customers.
- ▶ You can conduct business around the world more efficiently and provide better customer service and support all from a single platform.

1.4.8 Robust B2B and B2C

WebSphere Commerce is the industry's leading customer interaction platform that provides next-generation B2B e-commerce (B2B 2.0) capabilities that can help you redefine your strategy by:

- ▶ Streamlining and automating business processes to increase operational management efficiencies
- ▶ Optimizing sales and marketing effectiveness with buyer-centric marketing
- ▶ Strengthening relationships and customer satisfaction with a rich customer experience

Delivering a rich experience is fundamentally about becoming easier to do business with, from providing online access to catalog, pricing, and order

information to offering online communities to support customers post-purchase. The solution is to adopt proven technologies and concepts from the B2C world, such as Rich Internet Applications (RIA) and Web 2.0 technologies.

Buyer-centric B2B marketing can help you deliver targeted online offers and promotions tailored to customer segments. Increasingly, customers expect companies to fulfill their unique needs with a rich, online customer experience, whether it is B2C or B2B. WebSphere Commerce provides easy-to-use tools that enable marketing and merchandising managers to easily create and manage promotions, campaigns across multiple partner sites, or storefronts.

With a focus on the customer experience, WebSphere Commerce can help you give your B2B e-commerce clients a rich customer experience and help differentiate you from the competition. WebSphere Commerce offers capabilities to support transforming your online business into a next-generation B2B e-commerce experience. By focusing on operational management, buyer-centric marketing, and delivering a rich online customer experience, your B2B e-commerce strategy can help you reduce operational costs, increase sales, and deliver a differentiated online experience for your customers.

1.5 WebSphere Commerce Distributed Order Management

This section explains the high-level approach of integrating WebSphere Commerce with Sterling Distributed Order Management (DOM).

It is very common for a customer to already have an existing order management system or to want to use a third-party order management system to process their online orders captured from WebSphere Commerce. The external order management system will be responsible for processing the order, editing the order, and releasing the order to the appropriate fulfillment system. In most cases, inventory will also be managed by this external system. It is also very advantageous that a customer can leverage the same OMS system to manage orders, inventory, and fulfillment logistics across all their channels of sales.

WebSphere Commerce cross channel shopping: buy, fulfill, services from anywhere depicts an overview of today's cross channel solution, and the concept of buy, fulfill, and services can be performed from anywhere. As competition gets more intense, a seamless, unified, smart online customer interaction platform supported by an accurate, flexible, and efficient distributed order management and fulfillment logistics becomes more and more critical to business success (Figure 1-2).

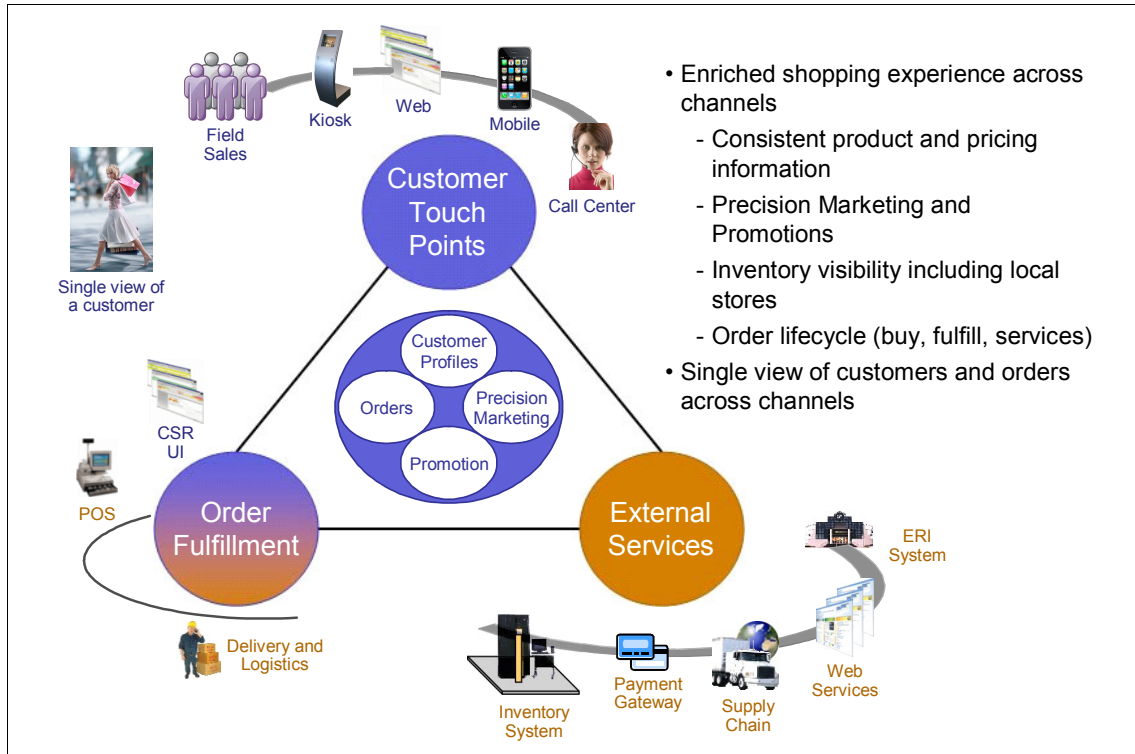


Figure 1-2 WebSphere Commerce cross-channel shopping: Buy and fulfill services from anywhere

Distributed Order Management integration

Commerce DOM integration enhances the shopping flow in a popular starter store called Madisons, while enhancing backend system integration with WebSphere Commerce. This integration provides comprehensive coverage of the order life cycle across channels, from capture to fulfillment.

A mandatory condition for enabling DOM (based on the value of the inventory type set at the store level) is the existence and resulting integration to an external inventory system. For this architecture, that external system is Sterling Selling

and Fulfillment Suite to provide the inventory support for WebSphere Commerce and the storefront.

The primary benefits and features provided by DOM include:

- ▶ Greater control over orders, regardless of their origin
Orders are stored in the Sterling central order management system, whether the product is purchased online, in-store, or through call centers. Order centralization enhances fulfillment options and provides greater order and supply visibility to improve fulfillment efficiency. The improved visibility enables Sterling Selling and Fulfillment Suite to prioritize supply requirements based on specific customer needs. Customer orders can be fulfilled by any channel to meet changing customer needs.
- ▶ Strong supply management functionality
Strong supply management functionality across channels enhances fulfillment abilities to help meet current customer needs. Sterling can automatically generate purchase orders when supplies are needed, while warning of events that could delay deliveries. Delays can be reduced by taking the appropriate actions, such as resubmitting an order or by routing the order to an alternate destination. In addition to providing order purchase capabilities, they can help speculate future supply demands based on current conditions.
- ▶ Accurate schedule monitoring and shipping requirements
Reduced overall costs are achieved by improving monitoring efficiency through accurate order tracking. High accuracy and efficiency enable faster delivery times and reduce overall storage and transportation costs.
- ▶ Useful reporting of key factors of the order life cycle
Improved monitoring capabilities enable generated reports focused on both business and customer needs. The reporting of key factors presents opportunities to modify and improve the current fulfillment model, ensuring future efficiency and customer satisfaction.

1.6 References

See the following resources for more information:

- ▶ “E-Commerce Platforms: A B2C Vendor Landscape,” AMR Research, 2009
- ▶ “Gartner Magic Quadrant for E-Commerce,” Gartner, Inc., July 2008



Sterling-Commerce solution overview

IBM Sterling Selling and Fulfillment Suite (SSFS) represents the best of breed for distributed order management. WebSphere Commerce is a leader in eCommerce solutions. The task to integrate the two is made considerably easier, as both leverage open standard integration technologies. With the release of WebSphere Commerce Version 7 Feature Pack 2, there are now out-of-the-box assets that provide real-time integration between the two.

This chapter covers the following topics:

- ▶ Solution overview
- ▶ WebSphere Commerce architecture
- ▶ WebSphere Commerce DOM cross-channel integration
- ▶ WebSphere Commerce integration architecture
- ▶ Enterprise Service Bus (ESB)
- ▶ Design considerations

2.1 Solution overview

WebSphere Commerce provides a powerful customer interaction platform for cross-channel commerce that can be used by companies of all sizes, from small businesses to large enterprises, and for many industries. While WebSphere Commerce delivers a smarter shopping experience that is seamless and integrated across all customer touch points, for some customers, the order management requirements have become too complex for WebSphere Commerce to handle by itself. Therefore, there is a need to integrate with a best-of-breed order management system (OMS) such as Sterling Commerce's Sterling Selling and Fulfillment Suite to handle such complex scenarios as Distributed Order Management, distributed inventory management, warehouse management, and so on.

Instructions to integrate WebSphere Commerce with a Distributed Order Management system have been available since WebSphere Commerce V6. WebSphere Commerce Version 7 Fix Pack 2 now provides the assets and methods to integrate with SSFS via a SOA-compliant ESB architecture to provide an integrated solution that encompasses the entire order life cycle.

2.2 WebSphere Commerce architecture

WebSphere Commerce provides a powerful customer interaction platform for cross-channel commerce that can be used by companies of all sizes, from small businesses to large enterprises, and for many industries. It provides easy-to-use tools for business users to create and manage precision marketing campaigns, promotions, catalog, and merchandising across all sales channels, allowing them to centrally manage a cross-channel strategy. WebSphere Commerce is a single, unified platform that offers the ability to do business directly with consumers (B2C), with businesses (B2B), indirectly through channel partners (indirect business models), or all of these simultaneously. WebSphere Commerce is a customizable, scalable, high-availability solution built to leverage open standards (Figure 2-1 on page 21).

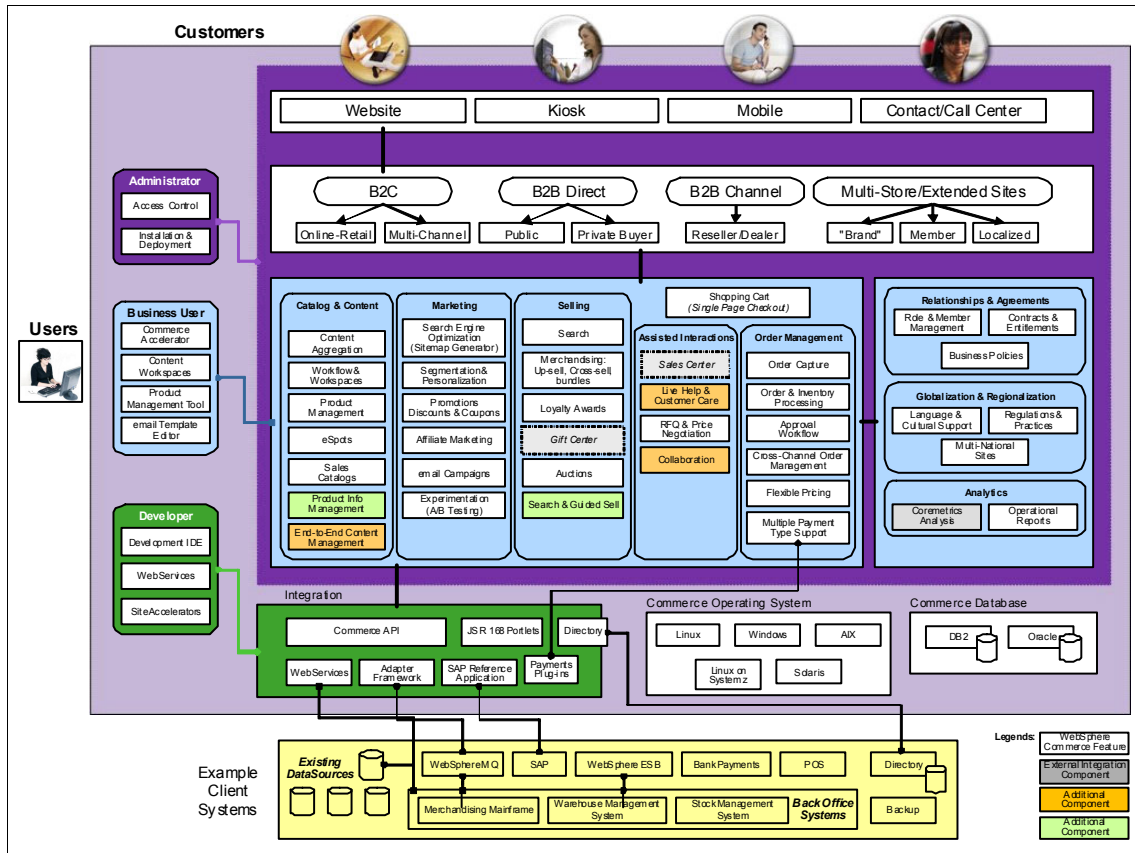


Figure 2-1 WebSphere Commerce architecture

2.2.1 Functional architecture

Functional architecture provides both the set of patterns used to implement the business functionality and the frameworks in which these business functions execute. The areas of business functionality of WebSphere Commerce include:

- ▶ Catalog
- ▶ Merchandising
- ▶ Marketing
- ▶ Promotions
- ▶ Trading
- ▶ User management
- ▶ Order management

2.2.2 Multi-channel presentation layers

WebSphere Commerce is multichannel-enabled, meaning that WebSphere Commerce can support transactions across various sales channels. The framework enhancements in this release support multiple presentation layers, responsible for displaying results, which decouple control logic from business logic (Figure 2-2).

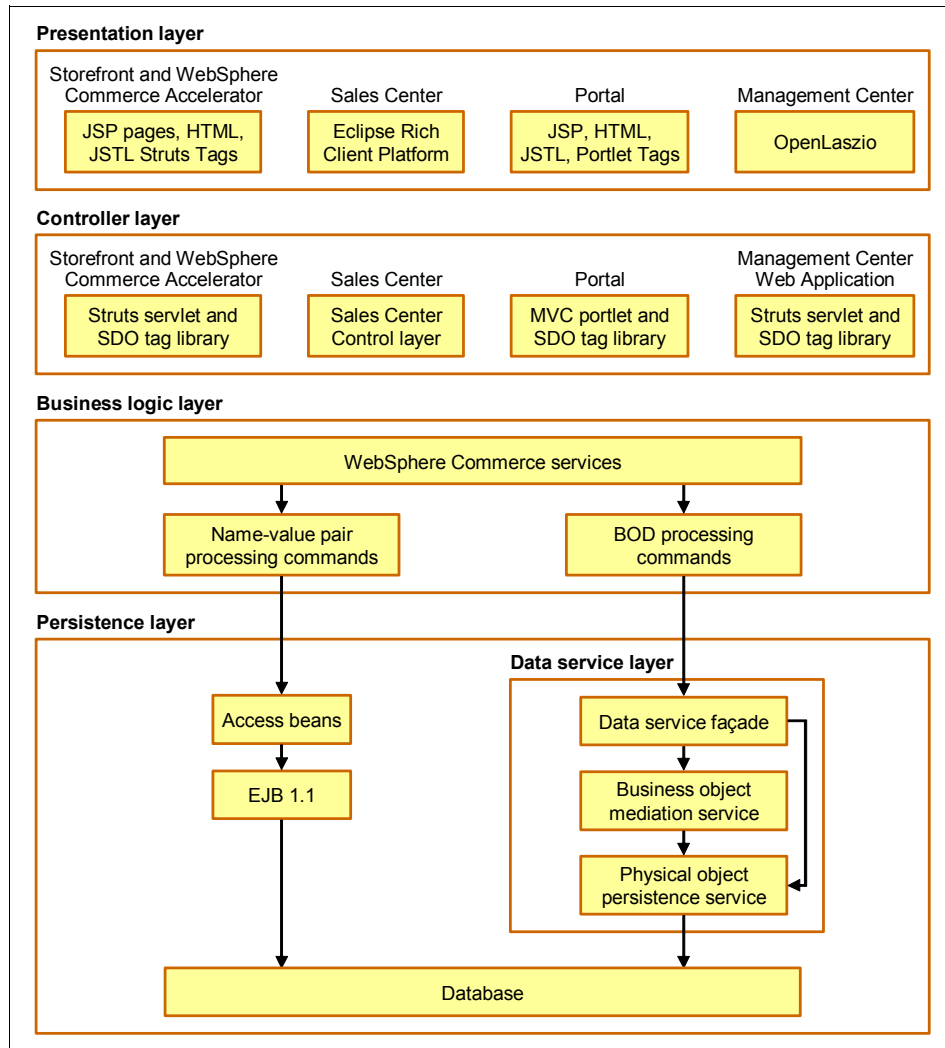


Figure 2-2 WebSphere Commerce multi-channel presentation layers

Figure 2-2 on page 22 depicts how WebSphere Commerce supports two channels:

- ▶ The web channel
- ▶ The sales channel

For the web channel the presentation is rendered using JSP pages, and the web controller layer uses Struts. For the sales channel, the display uses the Eclipse rich client technology. The presentation is rendered with Eclipse views and editors implemented using SWT components. Regardless of the channel, the business logic facade, a generic interface implemented as a stateless session bean, is used by controller calls to invoke controller commands. The command layer is implemented as WebSphere Commerce commands. The persistence layer provides EJB 2.0 support.

2.2.3 Application architecture

WebSphere Commerce comprises the following layers of the application architecture:

- ▶ Business models

In WebSphere Commerce, a business model represents a sample business situation in which the WebSphere Commerce product can be used. A business model describes a scenario in which various parties use WebSphere Commerce to achieve their needs. The five business models provided by WebSphere Commerce are:

- B2B direct
- Consumer direct
- Demand chain
- Hosting
- Supply chain

Within each business model, WebSphere Commerce provides one or more samples, referred to as starter stores, which may be used as a starting point to develop online sites. You can create other business models to suit your business needs.

- ▶ Presentation layer

The presentation layer is responsible for displaying results. By default, there are two supported types of presentation layers supported:

- Web
- Rich client

For the web presentation layer the display is rendered using JSP files, whereas, for the rich client the presentation is rendered with Eclipse views and editors implemented using SWT components.

► Service layer

The service layer, implemented using OAGIS messages, is a channel-independent mechanism that can access WebSphere Commerce business logic. The service layer segregates the implementation of business logic such as order and catalog. This segregation permits the underlying implementation to change without requiring that the caller change. All clients, including web clients and back-end services, go through the service layer to run business logic. The service layer supports two transport mechanisms:

- Local Java binding
- Web services

► Business logic

The business logic layer is where business rules are implemented independently of the presentation layer. Business logic is implemented using the command pattern. Two types of commands are implemented:

- Controller commands: Accessible by the presentation layer and used as a coordinator of tasks.
- Task commands: Not accessible by the presentation layer but called from the controller commands. This command type is used to implement business rules.

► Persistence layer

The persistence layer records the data and operations of the WebSphere Commerce system. The persistence layer represents entities within the commerce domain and encapsulate the data-centric logic required to extract or interpret information contained within the database. These entities comply with the Enterprise JavaBeans specification.

These entity beans act as an interface between the business components and the database. In addition, the entity beans are easier to comprehend than complex relationships between columns in database tables.

► Database schema

WebSphere Commerce database schema, which includes over 600 tables, is designed specifically for e-commerce applications and their data requirements. The database schema supports persistence requirements for the WebSphere Commerce subsystems (order, catalog, member, marketing, trading). WebSphere Commerce supports both DB2® and Oracle relational databases.

2.2.4 WebSphere Commerce framework overview

In WebSphere Commerce, the server run time defines the framework for handling system and user requests and performs the appropriate business logic to process the requests. The framework is built using an MVC design pattern and provides an environment that hosts business logic and handles persistence. It performs tasks such as transaction management and session management.

WebSphere Commerce and WebSphere support a variety of security mechanisms that can be used to protect access to data and other assets of the server. The access control framework in WebSphere Commerce prevents users from executing particular business logic. Not only does this access control framework provide a fine-grained control of business logic, it also provides the flexibility to restrict access to what data the user is allowed to view and modify. Access control allows you to group commands by access groups, and assign different customer commands to different owners, assign access to all owners, assign global site administrator access. By exploiting WebSphere global security, access to web resources such as servlets, JavaServer Pages (JSP) files, and Enterprise JavaBeans (EJB) methods can be controlled for an additional layer of security.

The command framework, an architectural component of the WebSphere Commerce server run time, provides the ability to execute commands that represent different business processes in the system. The command framework defines Java interfaces and abstract implementation that business logic extends and implements. Also provided is a set of base classes that commands can extend to simplify implementation.

Figure 2-3 shows the interactions between WebSphere Commerce components.

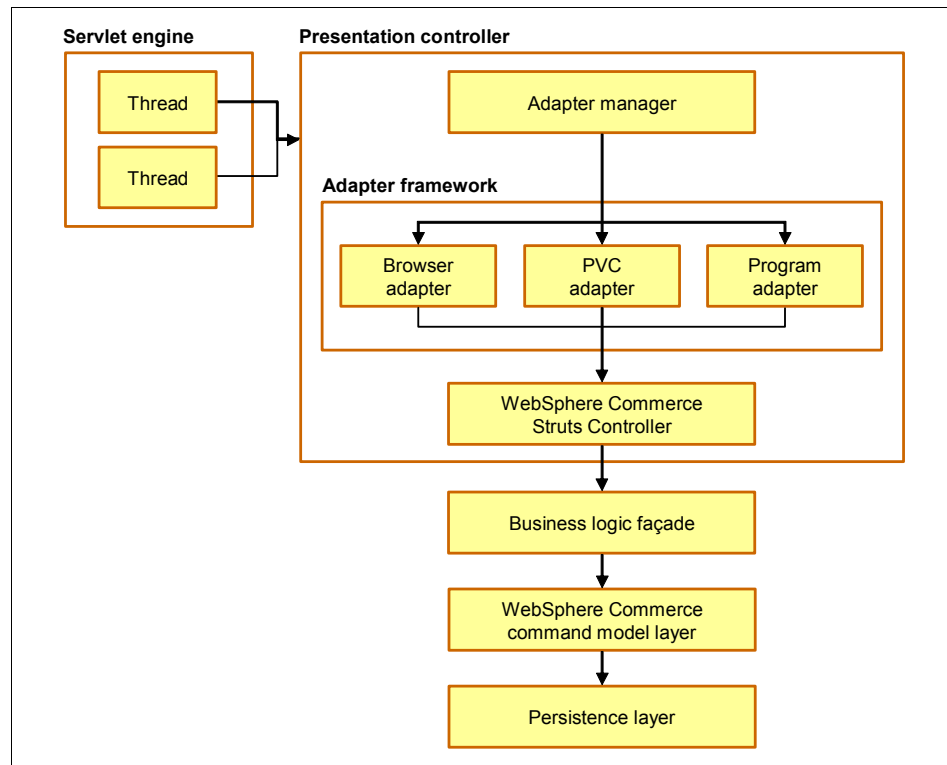


Figure 2-3 WebSphere Commerce framework overview

2.3 WebSphere Commerce DOM cross-channel integration

Since WebSphere Commerce Version 6 Feature Pack 5, DOM design has been part of a cross-channel integration solution specification. The overall objective of this design is to enhance the cross-channel integration capabilities of WebSphere Commerce in two areas:

- ▶ Shopping flow
- ▶ Commencing with WebSphere Commerce Version 7 Feature Pack 2, backend system integration to Sterling Selling and Fulfillment Suite (SSFS)

In terms of shopping flow, the focus of the WebSphere Commerce DOM design is to enable cross-channel shopping flows such as store location, stock location, buy-online-pick-up-in-store, buy-online-ship-to-store, and

reserve-online-pay-in-store. As for backend system integration, the focus of the design is to enable integration with SSFS, so as to provide a comprehensive coverage of the order life cycle, from capture to fulfillment, across channels.

To achieve this objective, WebSphere Commerce DOM provides:

- ▶ A store component with services for store location
- ▶ An inventory component with services to check inventory availability and process inventory requirements (for example, reservation) with caching and DOM integration capabilities
- ▶ Enhancements to the order component to support buy-online-pick-up-in-store and reserve-online-pay-in-store with DOM integration capabilities
- ▶ Enhancements to the new B2C starter store to demonstrate the buy-online-pick-up-in-store

The system context diagram shown in Figure 2-4 highlights the focus areas of this design.

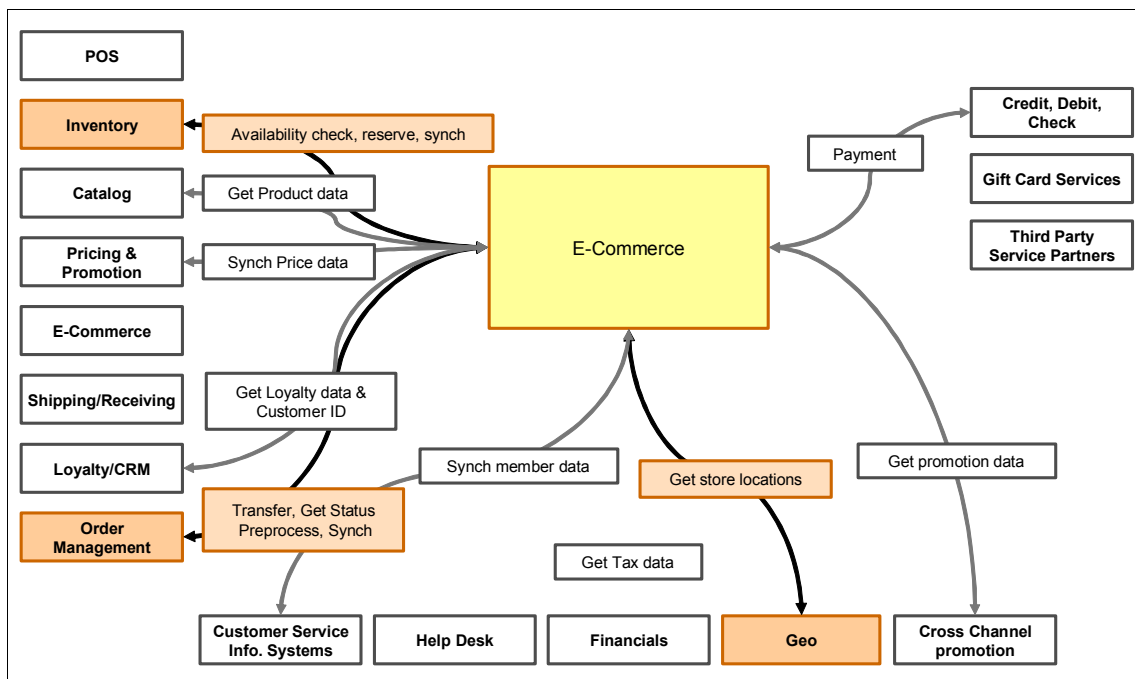


Figure 2-4 DOM system context diagram

DOM adheres to the latest WebSphere Commerce design guidelines in that its components and services are built as Commerce SOA components and services

utilizing the Business Object Document (BOD) programming model and web services framework. In addition:

- ▶ The DOM integration feature is compatible with the Extended Sites business model. See “Related publications” on page 243 for further references on Extended Sites.
- ▶ The DOM integration feature is configurable at the store level, as an alternative to the ATP and non-ATP inventory systems.
- ▶ The DOM integration feature offers the following advantages over caching inventory availability using the ATP and non-ATP inventory systems:
 - The DOM integration feature can cache ATP inventory availability, whereas the non-ATP inventory system can only cache non-ATP inventory availability.
 - It is much simpler to batch-load inventory availability to the DOM integration database schema, compared to the ATP inventory database schema.
 - The DOM integration feature supports in-memory caching in addition to database caching.
 - The DOM integration feature supports retrieving inventory availability from DOM in real-time when the information is not cached in WebSphere Commerce.
- ▶ Both the DOM integration feature and the ATP and non-ATP inventory systems will support buy-online-pick-up-in-store and reserve-online-pay-in-store.

DOM configuration should be used for any requirement where a dedicated Distributed Order Management system such as SSFS is to be integrated with Commerce.

Note: SSFS customers using the Commerce ATP or non-ATP inventory system methods should consider migrating to the DOM integration feature to take advantage of the improved integration flows and caching capabilities.

2.3.1 Commerce DOM integration detailed usage scenarios

Review this section for a more detailed analysis of WebSphere Commerce DOM integration usage scenarios.

Store location

In the case where the shopper wants to visit a retail location to pick up the desired products, an initial process is required. Before requesting inventory

availability, the shopper wants to find convenient locations that might be able to satisfy the request. These locations might be in relation to the shopper's home or workplace or stores that are near another location. The store location feature provided by DOM takes a location (for example, in the form of a postal/zip code) and shows relevant stores that are within the vicinity. The store location feature can be configured to return a predetermined number of stores within a predetermined distance from the originating location.

Stock location

The stock location feature allows retailers to remotely find available product inventory across the enterprise regardless of channel. Out-of-stock scenarios are frustrating to shoppers and can result in lost sales for retailers. Providing the ability to find available stock from various channels such as call centers, the web, or neighboring stores can help alleviate this problem. This solution provides a mechanism to remotely request availability for one or more products from an inventory system. This inventory system can be local to a store, specific to a channel, or common across the enterprise.

Availability requests specify one or more products and one or more locations to check for available inventory. The target inventory system specifies the availability for each of the products at each of the requested locations. The request can also include quantities required for each product. If a location does not have stock for all the products that are requested, it returns a partially completed listing of products and their availability. If no location has all the stock that is requested, the caller has the option of finding alternate locations that can satisfy the entire availability request.

Buy-online-pick-up-in-store/buy-online-ship-to-store

The option of buying or reserving online (or from one store) and picking-up in another store offers the shopper another option in the way in which he interacts with the retailer. The shopper can save shipping costs associated with a delivery and can choose a time that is convenient to pick up the merchandise rather than having to wait for a delivery. By enabling this option, the retailer provides this convenience option to their shoppers and also has the benefit of directing the shopper to the store, where additional purchases may take place.

Reserve online pay in-store

Reserving online and picking up in a store requires a shopper to find desirable merchandise through the web channel and then choose to pick up the items in a retail store. Alternatively, if a shopper is browsing in a retail store that is out of stock of a particular item, the shopper can reserve this item in another store. In the reserve case, the shopper does not pay for the products immediately, but rather asks for the merchandise to be set aside in a particular store so that he can subsequently visit the store to pick up the products and pay for them there.

In the buy scenario, the situation is similar except the payment processing is initiated from the original channel. The shopper chooses to pick up the pre-authorized merchandise at a particular store. When the shopper visits the store, if a suspended POS transaction was created, it can be resumed so that the transaction can be completed.

Both of the above scenarios require the stock location feature to locate and validate available inventory at selected stores.

2.3.2 Reliability and performance strategy

In integration scenarios where Commerce has to exchange information with one or more backend systems, the ultimate goal in terms of reliability and performance is to reach a level of performance, scalability, and availability comparable to processes that are entirely local. However, this is often hampered by the following factors:

- ▶ Performance of the backend system: Can the backend system respond to real-time requests in a timely manner, or is it more designed for batch processing?
- ▶ Scalability of the backend system: Can the backend system handle the amount of traffic generated by the web channel during peak periods, or is it more designed for back office traffic?
- ▶ Availability of the backend system: Is the backend system designed with high availability in mind, or does it require regular down times for maintenances? Is the network connection to the backend system reliable?

2.3.3 Stock location

In the stock location scenario, SSFS is the system of record of inventory availability. To address considerations for performance, scalability, and availability, the inventory component and its services are designed with caching support. More specifically, the inventory availability of an item at a location can be cached by the component, either in memory, in the database, or both.

2.3.4 Transfer order service requests

The transfer order service requests can be configured as asynchronous and routed to a message queue instead of directly to DOM. This way DOM can batch-process the transfer order service requests instead of handling them in real time.

2.4 WebSphere Commerce integration architecture

In this section, we discuss the WebSphere Commerce DOM architecture for integrating with SSFS.

2.4.1 Built-in DOM store and inventory integration

WebSphere Commerce, together with SSFS, enables sellers to interact with their customers by providing end-to-end functions to take and manage orders. WebSphere Commerce is the selling engine, displaying products and providing marketing and promotional tools and features to lead a customer to place an order. Once the order is placed, SSFS manages the order and inventory through the various components required to fulfill that order.

Message interactions from WebSphere Commerce to SSFS shows the types of interactions available from WebSphere Commerce to SSFS, finally, to the ERP systems, as demonstrated in Figure 2-5.

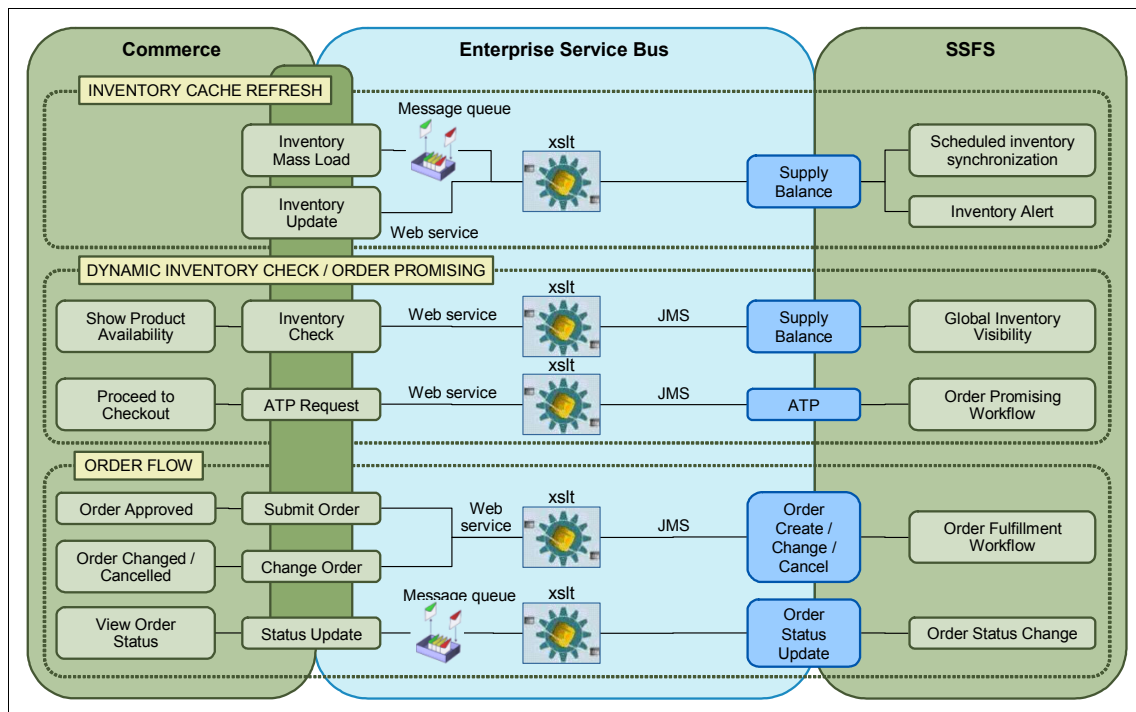


Figure 2-5 Message interactions from WebSphere Commerce to SSFS

The SSFS to WebSphere Commerce integration architecture implements an Enterprise Service Bus (ESB), which is explained further in 2.5, “Enterprise Service Bus (ESB)” on page 36. Refer to the WebSphere Commerce Info Center for further discussions on how WebSphere Commerce integrates with external systems.

Searching in WebSphere Commerce Info Center V7 for DOM will provide links that describe WebSphere Commerce's business functions and integration services to SSFS that will provide improved overall end to end business functions using out of the box functionality. Important business functions provided that enrich the Commerce Order Capture process are:

- ▶ Store component, which brings in the concept of the multiple physical stores and the attributes for each store. Examples of individual store's attribute and business functions are:
 - Defining the logical model of physical stores and geographical regions
 - Providing services to locate physical stores by geocode and distance, geographical region, or store attribute
 - Locally storing physical store data within the WebSphere Commerce database with no need to upload any data to the mapping service provider
 - Integration support for searching by address, map display, and driving directions with mapping service providers (for example, Google Maps, MapQuest, Yahoo Maps)

A detailed mapping service provider integration example can be found in *Building Multi-channel Applications with WebSphere Commerce*, SG24-7787. See “Related publications” on page 243.

- ▶ Inventory component, which:
 - Defines the logical model of inventory availability and inventory requirements
 - Provides services to retrieve the inventory availability of products online and in-store and process inventory requirements (that is, reserve inventory)
 - Can delegate requests to SSFS via the ExternalInventoryFacadeClient
 - Can cache inventory availability locally using the WebSphere Commerce inventory cache

The inventory availability master information typically resides in the responsibility of backend inventory management systems (SSFS). However, some backend inventory management systems might have performance, scalability, and availability limitations. The inventory component and its services are designed with caching support to address these limitations. That is, the inventory availability of an item at a location

can be cached by the component, either in memory or in the database, depending on configuration.

An inventory cache might become an important consideration to page browsing performance if the retailer has a large product assortment and number of store locations and the costs of importing all inventory availability records into the database outweighs the savings from caching the information. During the checkout process, when the shopper has decided to place the order, the specific inventory cache and inventory data in DOM could be updated accordingly.

- ▶ Order component:
 - Order logical model, commands, and services updated to support checkout flows that tie an order to a specific physical store's inventory managed by SSFS, and other attributes for the physical store (for example, buy-online-pick-up-in-store).
 - New payment method to support pay in-store.
 - Transferring of submitted orders to SSFS via the ExternalOrderFacadeClient.
 - SSFS can sync (push) order statuses to WebSphere Commerce using the order component's SyncOrder service.

2.4.2 WebSphere Commerce DOM integration specifics

As of WebSphere Commerce Version 6 Feature Pack 5, generic DOM integration is part of the cross-channel integration solutions. It is a back-end system integration that, as of WebSphere Commerce Version 7 Feature Pack 2, provides pre-built assets that support integration to SSFS built with WebSphere Commerce DOM integration components.

Inventory management

WebSphere Commerce DOM integration provides a new inventory component with SOA services to check inventory availability and process inventory requirements, including caching and DOM integration capabilities.

Cache management

There is greater flexibility in the inventory cache, which can use a distributed object cache mechanism in WebSphere Application Server or WebSphere Commerce databases through different configurations.

In the WebSphere Application Server distributed object cache mechanism, the DOM inventories are cached in the memory. The memory cache can serve the

inventory request with a quicker response. In a cluster environment, the inventory caches are synchronized between different cluster members.

With the WebSphere Commerce database cache, the DOM inventory records are resident in DB2 databases or Oracle databases. WebSphere Commerce has supplied an efficient approach to access these databases by way of a data service layer (DSL).

There is an extended version of the dynamic cache monitor, which can help monitor the distributed object caches.

Whether the local DOM inventory cache is in memory or in database, it is configured in the WebSphere Commerce database. Store administrators can change these configurations to change the cache locations.

See “Related publications” on page 243 to for references for WebSphere Commerce caching and DSL.

Note: There are other caching strategies not used by DOM, but that are necessary for other non-functional requirements. They are:

- ▶ Cachespec.xml directives with servlet caching enabled
- ▶ Command caching
- ▶ Registry caching
- ▶ Web services caching

Store location management

A new store component with services for store location gives WebSphere Commerce online shoppers the capability to find nearby stores based on entered locations. The shoppers can then pick up merchandise after they shop online or can go to the stores directly for shopping.

The enhancement of catalog browsing enables the shoppers to view product availability for the certain stores. Before a shopper can see the availability, shoppers need to use Store location feature to select their favorite stores. They can also select their favorite stores in the product detailed information display page.

2.5 Enterprise Service Bus (ESB)

This section introduces the concept of the Enterprise Service Bus as a subcomponent of the integration layer between SSFS and WebSphere Commerce.

2.5.1 IBM SOA Reference Architecture

An ESB is a middleware infrastructure component that supports the implementation of service-oriented architecture (SOA), which is deployed and leveraged to provide support and binding between the SSFS and WebSphere Commerce disparate integration technologies.

The IBM SOA Reference Architecture defines the IT services required to support an SOA. It includes development environment, services management, application integration, and runtime process services. The capabilities of the architecture can be implemented on a build-as-you-go basis as new requirements are addressed over time.

Figure 2-7 shows the IBM SOA reference architecture and the supporting software.

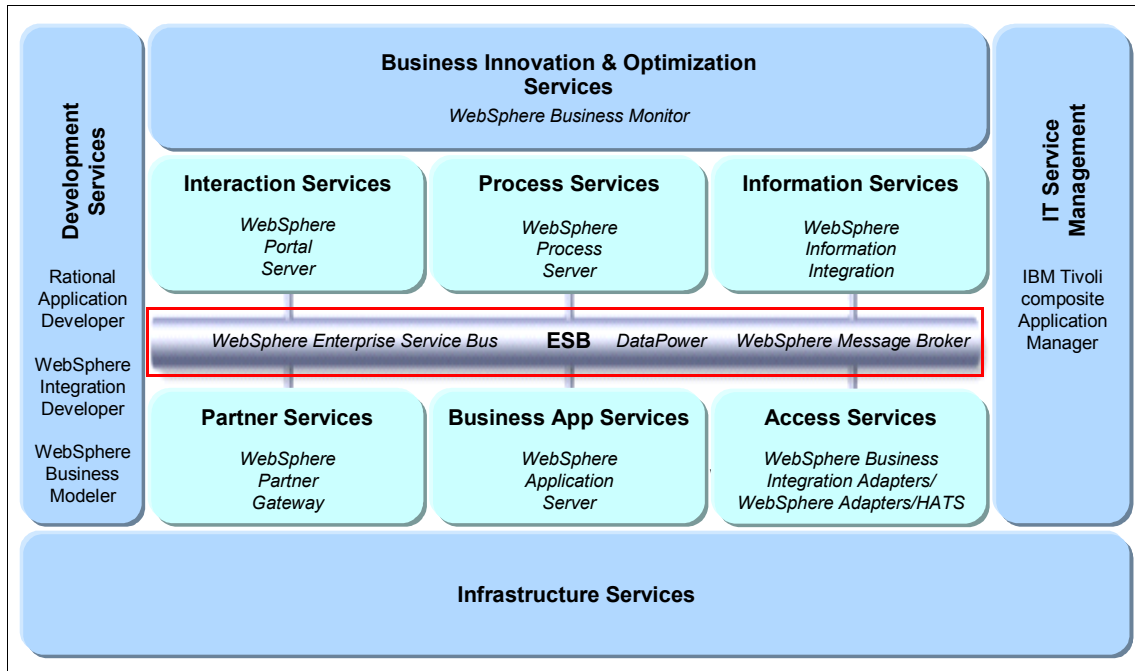


Figure 2-7 IBM SOA reference architecture with product mapping

There are three IBM products that can provide the ESB capabilities for SSFS and WebSphere Commerce integration:

- *WebSphere Enterprise Service Bus (WESB)* provides ESB functions for SOAs built on open standards. It is based on WebSphere Application Server Network Deployment and inherits its built-in messaging provider and quality of services. WebSphere Process Server (WPS) is built on top of WebSphere Enterprise Service Bus and adds a business process run time.

Note: WESB is the technology leveraged in WebSphere Commerce Version 7 Feature Pack 2.

- *WebSphere Message Broker (WMB)* provides advanced ESB functionality for universal support of messaging applications. It is based on WebSphere MQ and takes advantage of the services that are provided by its messaging infrastructure.
- *WebSphere DataPower® SOA Appliances* are a key element in the IBM holistic approach to SOA. These appliances are purpose-built, easy-to-deploy

network devices to simplify, help secure, and accelerate your XML and web services.

2.5.2 SSFS and WebSphere Commerce integration via ESB

SSFS and WebSphere Commerce both have robust and flexible integration capabilities. What is provided out-of-the-box is SOA compliant and based on industry standards. For messaging requirements, SSFS leverages Java Messaging Services (JMS), and WebSphere Commerce leverages web services. Web services define a standard way of describing the interface to a service provider that allows some level of decoupling (as the actual implementation details are hidden). However, web services imply a direct connection between the consumer and provider. Bridging these two integration technologies (JMS/web services) can be a challenge.

Fundamentally, an ESB is a flexible connectivity infrastructure for integrating applications and services. The SSFS/WebSphere Commerce ESB addresses the following:

- ▶ The ESB manages the bridging between the communication transport protocols. SSFS implements the JMS API over a bus-oriented transport layer, and WebSphere Commerce leverages web services over HTTP. The ESB handles the bridging without either end point being aware of the disparate technologies.
- ▶ SSFS and WebSphere Commerce need only be aware of each respective ESB endpoints for routing requirements, thus decoupling the two. Either system in a consumer role binds to the intermediary ESB to access a service provided by the other, with no direct coupling to the actual provider of the service. The intermediary ESB maps the request to the location of the real service implementation.
- ▶ There are considerable mapping requirements for each transaction associated with order and inventory management, as each system has its own view of those respective entities. The overhead and transforming message formats are isolated and maintained in the ESB.
- ▶ Authorize/authentication of the message payload is based on requirements.
- ▶ A degree of reliable messaging is provided via the bus supporting the JMS queues. The ESB can extend the reliability of the JMS queue to the web service transaction.

2.5.3 WebSphere Enterprise Service Bus

The assets provided by WebSphere Commerce Version 7 Feature Pack 2 are built for WebSphere Enterprise Service Bus (WESB). For integration requirements not provided by this release, you can follow a detailed step-by-step process and sample code. Refer to the Building a WebSphere Enterprise Service Bus mediation module for DOM Integration topic in the WebSphere Commerce Version 7 Information Center:

<http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.dom-integration.doc/tasks/tsmdombuildmedmod.htm>

WESB delivers an Enterprise Service Bus (ESB) infrastructure to enable connecting applications that have standards-based interfaces (typically a web service interface described in a WSDL file). It provides mechanisms to process request and response messages from service consumers and service providers that connect to the ESB.

WESB is the mediation layer that runs on top of the transport layer within WebSphere Application Server. As such, WESB provides prebuilt mediation functions and easy-to-use tools to enable rapid construction and implementation of an ESB as a value-add on top of WebSphere Application Server (Figure 2-8).

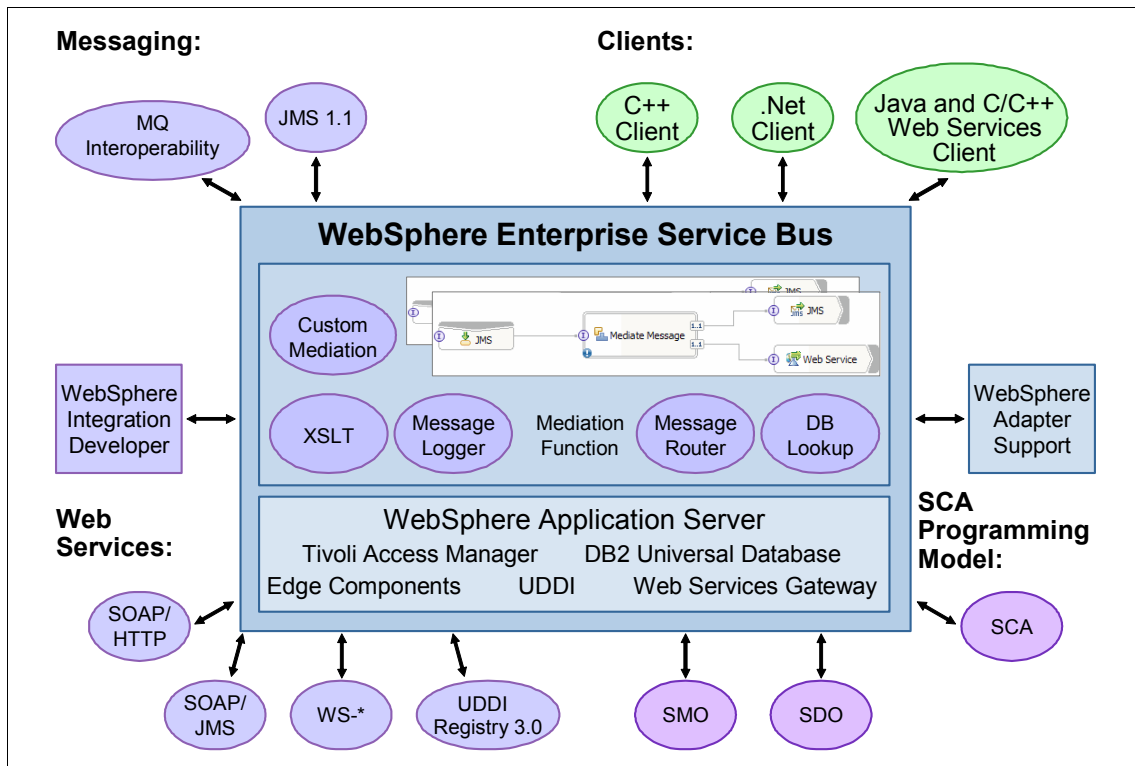


Figure 2-8 WebSphere Enterprise Service Bus

WESB leverages WebSphere Application Server Network Deployment qualities of service, with its clustering, failover, scalability, security, and a built-in messaging provider.

To summarize the ESB introduction, WESB adds the following value to the application server:

- ▶ It provides built-in mediation functions, which can be used to create integration logic for connectivity.
- ▶ It supports the SOA-based standard programming model, Service Component Architecture (SCA), which enables rapid development of mediation flow components.

- ▶ WebSphere Integration Developer (WID) is an easy-to-use development tool that supports WebSphere Enterprise Service Bus.
- ▶ It leverages WebSphere Application Server. WESB offers JMS messaging and WebSphere MQ interoperability for messaging, in addition to a comprehensive clients package for connectivity.
- ▶ It offers support for J2EE Connector Architecture based WebSphere Adapters.

WESB also is part of the infrastructure of the Common Event Infrastructure (CEI), which is the foundation for monitoring applications. IBM uses CEI throughout its product portfolio, and for monitoring products from Tivoli®. The event definition (Common Business Event) is standardized through the OASIS standards body, so that other companies and customers can use the same infrastructure to monitor their environments.

2.5.4 WebSphere Integration Developer

WebSphere Integration Developer (WID) is the graphical development environment for developing end-to-end WESB integration between Sterling Selling and Fulfillment Suite (SSFS) and WebSphere Commerce. Business functions are encapsulated into service components that are assembled to create an integrated application, which is also a service. The services created comply with the leading industry-wide standards.

WID allows both a top-down design approach to building an integrated application, where the components are assembled together (and implementation is added later) and a bottom-up approach, where the components are already implemented and you assemble them in a visual editor and then create a logical flow amongst them by joining them with lines. The Commerce Version 7 Feature Pack 2 assets were built in this way. In addition, these assets can be tested in a WID debugging and test environment with the ability to monitor in real time to fine tune them for optimal performance.

For custom development, refer to “Building a WebSphere Enterprise Service Bus (WESB) mediation module for DOM integration” at:

<http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.dom-integration.doc/tasks/tsmdombuildmedmod.htm>

2.5.5 WebSphere Message Broker

Sterling Selling and Fulfillment Suite (SSFS) and WebSphere Commerce can leverage other IBM integration technologies such as IBM WebSphere Message Broker (WMB). WebSphere Message Broker provides an alternative for an

integrated ESB that can connect just about any application or service to any other application or service, which can be particularly beneficial for large heterogeneous environments. WebSphere Message Broker handles a broad range of transports and protocols, including WebSphere MQ, Java Message Service (JMS) Version 1.1, Hypertext Transfer Protocol over Secure Socket Layer (HTTPS), web services, file, and user-defined protocols. It handles practically any data format.

Figure 2-9 shows the WebSphere component architecture.

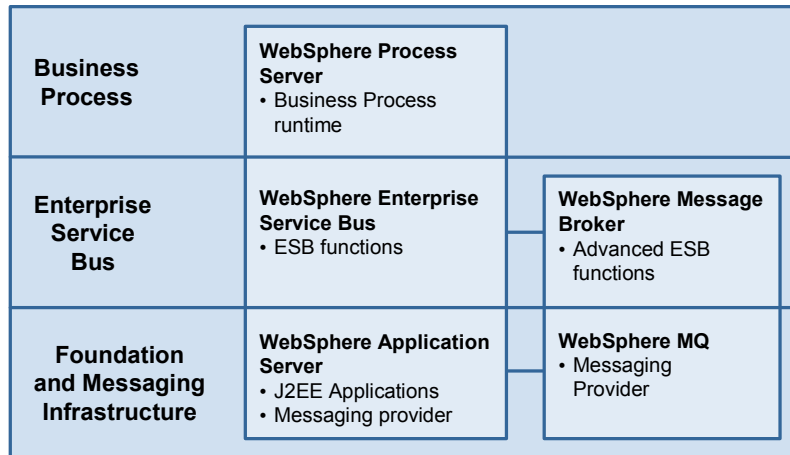


Figure 2-9 WebSphere Message Broker/MQ component architecture

WebSphere Message Broker enhances the flow and distribution of information by enabling the transformation and intelligent routing of messages without the need to change either the applications that are generating the messages or the applications that are consuming them. WebSphere Message Broker is a set of application processes that host and run message flows consisting of a graph of nodes that represent the processing needed for integrating applications. WebSphere Message Broker processes an inbound message before passing it on to one or more other business applications. WebSphere Message Broker routes, transforms, and manipulates messages according to the logic that is defined in message flow applications.

A key component is WebSphere MQ (WMQ) as the transport mechanism for WebSphere Message Broker. WebSphere Message Broker has a distributed architecture, and each broker has a database in which it stores the information that it needs to process messages at run time.

Load balancing and high availability can be achieved by providing multiple broker instances serving the same logical hub where each instance is mapped to its

own WebSphere MQ queue manager. The different broker instances could reside on different machines.

Therefore, due to the store and forward (and publish and subscribe) capability of WebSphere MQ, the persistence nature of the broker databases, and load balancing options, the combined product set is a very robust ESB with considerable reliable messaging capability via its high availability features.

You can find more information about IBM WebSphere Message Broker at the WebSphere Message Broker home page:

<http://www-01.ibm.com/software/integration/wbimessagebroker/>

You can find an example on how to implement Sterling-Commerce with Message Broker as the mediation layer by reading Chapter 3.3 Implementation of WebSphere Message Broker mediation module for DOM integration in *Building Multichannel Applications with WebSphere Commerce*, SG24-7787.

2.5.6 DataPower

WebSphere DataPower SOA Appliances are a key element in IBM's holistic approach to service-oriented architecture (SOA). These appliances are purpose-built, easy-to-deploy network devices to simplify, help secure, and accelerate your XML and web services. IBM's hardware ESB, the XI50 and XI50B, are purpose-built for simplified deployment and hardened security, bridging multiple protocols and performing conversions at wirespeed.

DataPower appliances have excellent coverage for these SOA tenants for an ESB, as they:

- ▶ Support the abstraction view with its proxy/gateway architecture
- ▶ Are highly message oriented, with thorough support for SOAP, raw XML, and unprocessed (binary) documents
- ▶ Have full web services that support the correct granularity
- ▶ Are network aware, residing between the network and application layer with full support for message-based routing, transformation, protocol mediation, filtering, and extensive security

DataPower possesses all the major services that are required to implement an ESB:

- ▶ Transformation
- ▶ Routing
- ▶ Validation
- ▶ Authentication
- ▶ Logging
- ▶ Protocol bridging

SSFS WebSphere Commerce implementation

As the mediation component, DataPower has native web services support for interfacing with Commerce. DataPower functions as a web services protocol bridge to communicate with SSFS. To bridge to JMS, DataPower uses the IBM JetStream Formats and Protocols (JFAP) to connect to the WebSphere JMS object in the underlying WebSphere Application Server, which provides integration services to SFSS.

Figure 2-10 shows the protocol bridging method of DataPower for Sterling-Commerce.

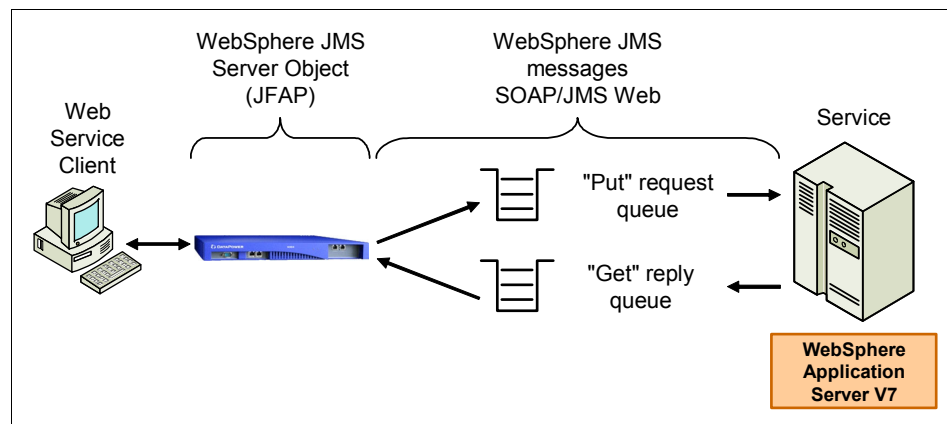


Figure 2-10 DataPower protocol mediation layer

2.6 Design considerations

This section discusses design considerations for integrating SSFS with WebSphere Commerce end-to-end.

Other integration requirements

An enterprise should integrate SSFS with WebSphere Commerce via the WebSphere Commerce DOM framework. But that framework does not support integration requirements for other business objects that do not necessarily require near real-time updates, so point-to-point integration technology is provided that encompasses the entire order life cycle.

The overarching integration requirements will be based on which system maintains the objects (that is, which is the system of record). Business requirements drive system ownership, taking into account such things as legacy constraints. Each enterprise has its own set of requirements. There is no firm model for system ownership, but there are practical considerations that usually give a good idea of which system owns what.

Table 2-1 provides a straw man view of system ownership.

Table 2-1 System of record

Component	Business entity	System of record
Member	User	WebSphere Commerce
Member	User group/role	WebSphere Commerce/Sterling
Member	Customer	WebSphere Commerce
Member	Buyer organization	WebSphere Commerce
Catalog	Catalog data	third party catalog management system
Contract	Account	WebSphere Commerce
Contract	Contract	WebSphere Commerce
Marketing	Campaign	WebSphere Commerce
Marketing	Promotion	WebSphere Commerce
Inventory	Inventory data	Sterling
Pricing	Pricing data	WebSphere Commerce
Payment	Payment data	WebSphere Commerce
Order capture	Shopping cart	WebSphere Commerce
Order capture	Requisition list	WebSphere Commerce

Component	Business entity	System of record
Order capture	Quote	WebSphere Commerce
Order processing	Order	Sterling
Order modification	Order	Sterling
Order fulfillment	Order	Sterling
Return	RMA	Sterling



Business scenarios

This chapter explains the basic scenarios for the business users and for end users. The scenarios in this chapter explain how to easily create and manage business components such as store, catalog, categories, products, items, inventory, and orders. These scenarios involve participants such as administrator, user, and store manager. These scenarios explain the actions and the expected behaviors for each flow.

This chapter contains the following sections:

- ▶ 3.1, “Business scenarios overview” on page 48
- ▶ 3.2, “SSFS WebSphere Commerce scenarios” on page 48
- ▶ 3.3, “Available features in WC” on page 55
- ▶ 3.4, “Scenarios covered OOB” on page 56

3.1 Business scenarios overview

An end-to-end shopping flow involves a number of interactions between two major systems, the WebSphere Commerce and the Sterling Commerce. WebSphere Commerce acts as the customer interaction and order capture layer of the shopping experience, whereas all the order component-related transactions involve Sterling Commerce. When a customer browses the catalog, all the catalog and product information are retrieved from WebSphere Commerce and the inventory is pulled from Sterling Commerce. All order manipulations, such as adding an item to a cart, managing ship-to and bill-to addresses, entering payment information, and qualifying the customer and order for appropriate marketing and promotions are operated still from WebSphere Commerce, with necessary interactions with Sterling Commerce.

Throughout this process, WebSphere Commerce retrieves the product inventory either from its own inventory cache or from the Sterling Commerce Sterling Distributed Order Management (DOM) to inquire about warehouse or local store inventory. These options can be set based on the specific customer requirements. For certain scenarios, WebSphere Commerce also needs to retrieve order status information from the Sterling Commerce DOM database.

Once the shopper completes shopping and submits the order, WebSphere Commerce transfers the order to Sterling Commerce, which then manages all aspects of the order fulfillment process. WebSphere Commerce act as the front end for order creation, update, and verification of the order status. Actual order processes, such as how inventory is allocated and reserved, how orders are split and fulfilled, order state change from one status to the next, and any other manipulations of the order, are in the domain of the Sterling Commerce system.

3.2 SSFS WebSphere Commerce scenarios

This section discusses typical order shopping flow overviews that leverage the out-of-the-box integration between Sterling Selling and Fulfillment Suite (SSFS) and WebSphere Commerce. WebSphere Commerce Madisons Starter Store is deployed with this functionality, and the various scenarios are captured below.

Note: These examples are a subset of a typical SSFS-WebSphere Commerce implementation. See both products' documentation for the remaining integration capability between the two products.

3.2.1 Order browse

The out-of-the-box WebSphere Commerce sample online store displays one or multiple selected stores' inventory only after a shopper has decided and selected a physical store from which to pick up the order. This approach provides better scalability to support a large number of physical stores.

The order browse detail (Figure 3-1) steps are:

1. Find and set the preferred store (optional).
2. Browse the catalog.
3. Get the inventory availability.

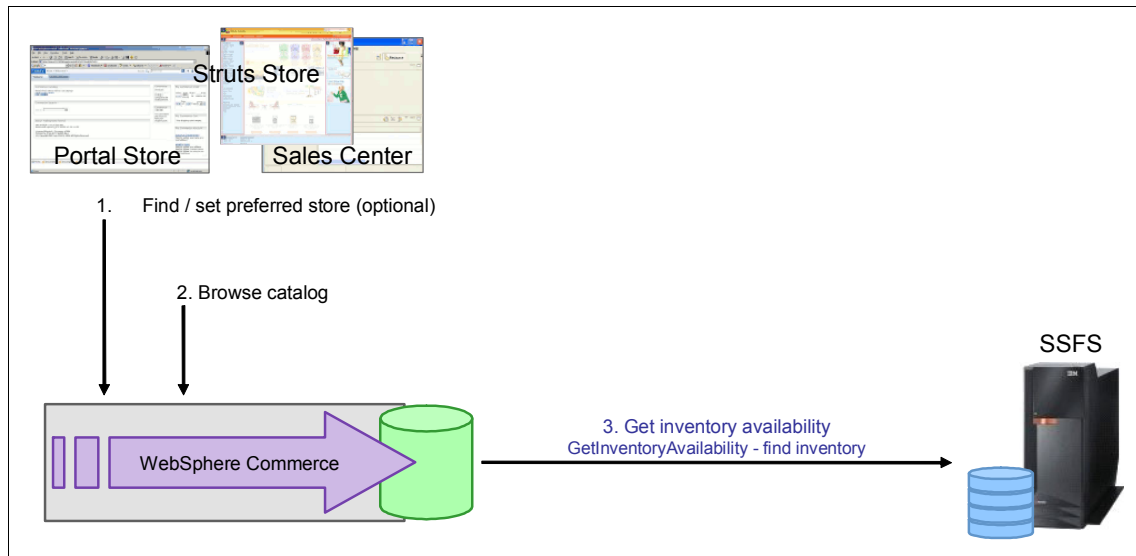


Figure 3-1 Order browse flow

3.2.2 Order capture

After catalog browsing, once the shopper is ready with his shopping cart, he can view the shopping cart. This is where the actual inventory of those items will be locked. The order information is packaged by WebSphere Commerce and sent to the Sterling Selling and Fulfillment Suite (SSFS) system, which fulfills the order and provides the status throughout the order life cycle.

The shopping flow (Figure 3-2) comprises the following steps:

1. The online shopper browses the catalog and finds a product at her favorite store.
2. WebSphere Commerce interacts with SSFS to obtain the inventory availability of the product at the shopper's favorite store.
3. The shopper adds the product to the shopping cart.
4. WebSphere Commerce interacts with SSFS to check whether there is sufficient inventory.
5. The shopper checks out.
6. WebSphere Commerce interacts with SSFS to reserve inventory for the order.
7. The online shopper submits the order.
8. WebSphere Commerce transfers the order to SSFS.

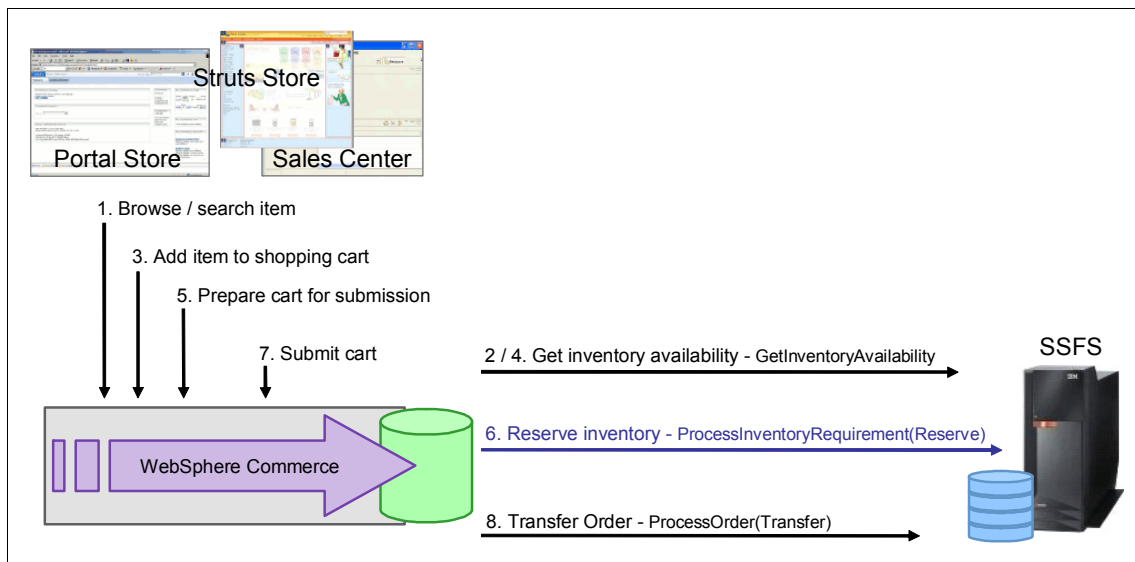


Figure 3-2 Shopping flow in SSFS-Commerce integration solution

3.2.3 Order status

After an order has been placed in WebSphere Commerce, SSFS and possibly other external systems deal with the delivery of that order. All aspects of fulfilling the order (whether a matter of packing items from inventory or manufacturing items and all aspects of billing, shipping, and so on) are handled by the external systems. Because the external systems manage an order's status at a more granular level,

the WebSphere Commerce's picture of that order must be updated with the order's status as it moves through the external systems in order to give an accurate representation of the status.

There are two possible scenarios with order status. One is where order status is requested from within WebSphere Commerce (Figure 3-3), and the other is where the order status is pushed to WebSphere Commerce. WebSphere Commerce maintains a cache of the latest order statuses, which is updated by WebSphere Commerce during the many steps of order capture processing. After the order is submitted to SSFS for processing, the order status can be updated by using data received from SSFS.

Note: The choice of using the push or pull approach of updating the order status is one of the enterprise architecture decisions.

Figure 3-3 shows an order status request from WebSphere Commerce.

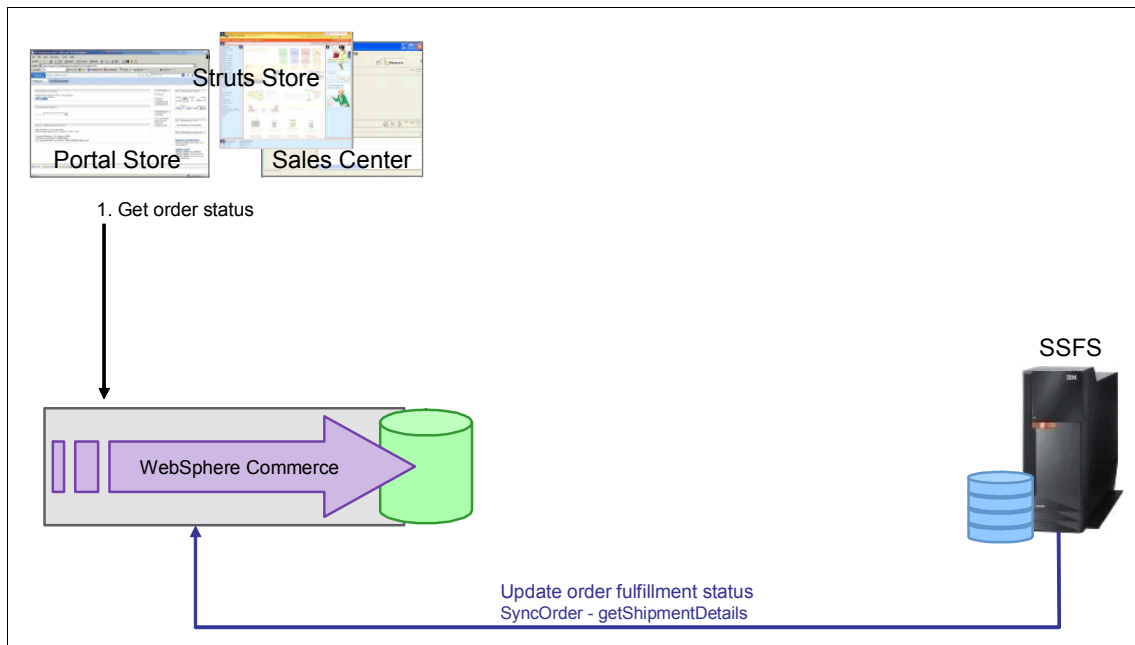


Figure 3-3 Order status flow

3.2.4 Order cancel

A shopper can cancel an order if the order is in a state that allows cancellation. Once an order has been sent to an external system for fulfillment, the order goes through

a number of transactions, such as picking, packing, and charging the payment method. At some point in this process, the order reaches a status after which it cannot be cancelled.

Figure 3-4 shows the detailed steps for canceling an order from WebSphere Commerce.

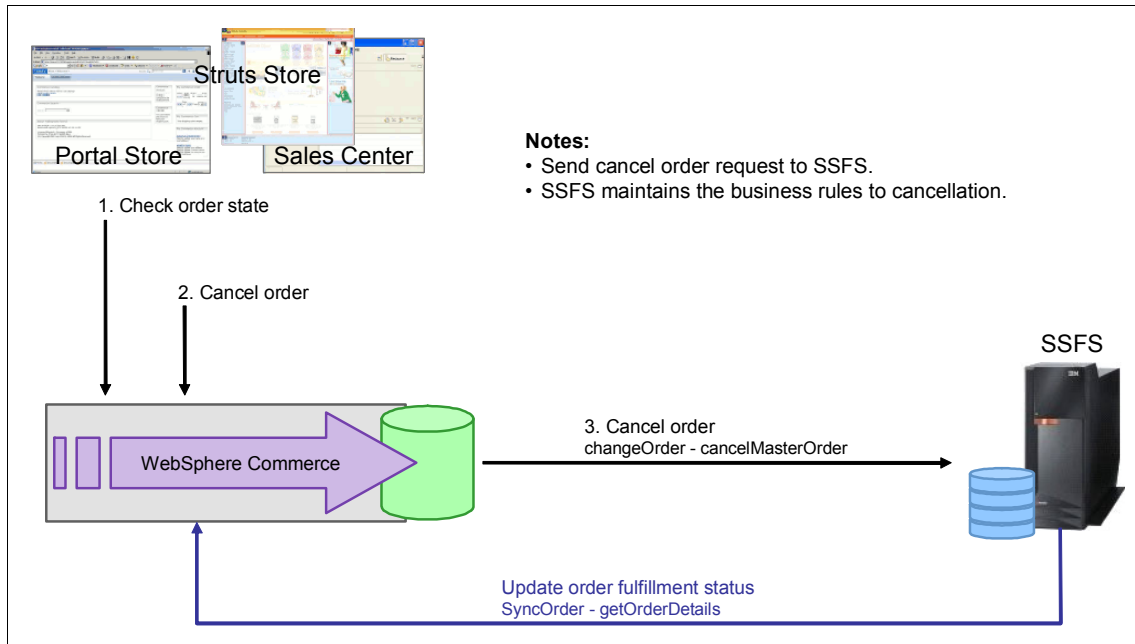


Figure 3-4 Order cancel flow

The order cancel steps are:

1. Check the order state.
2. Cancel the order from WebSphere Commerce.
3. The cancel order request is sent to SSFS with a status of returned to WebSphere Commerce.

3.2.5 Order modification

A shopper can modify an order in a number of ways, either related to the order information or to the order item information. WebSphere Commerce allows, through the use of the Accelerator (a tool provided out-of-the-box with WebSphere Commerce to be used by internal resources, typically in a customer

service type of role), the following modifications to an order (additional order changes can be supported but might require customization):

- ▶ Changing the quantity of products in an order
- ▶ Changing the purchase order number for an order in a B2B direct store
- ▶ Adding a product to an order
- ▶ Removing a product from an order
- ▶ Selecting another shipping address for an order
- ▶ Selecting another shipping method for an order
- ▶ Changing the total price of an order
- ▶ Editing an order level adjustment
- ▶ Changing the payment options for an order
- ▶ Selecting another billing address for an order
- ▶ Adding a comment to an order

Note: As with order cancellation, it is a business decision as to when an order can or cannot be modified.

Figure 3-5 shows the detailed steps for modifying an order from WebSphere Commerce.

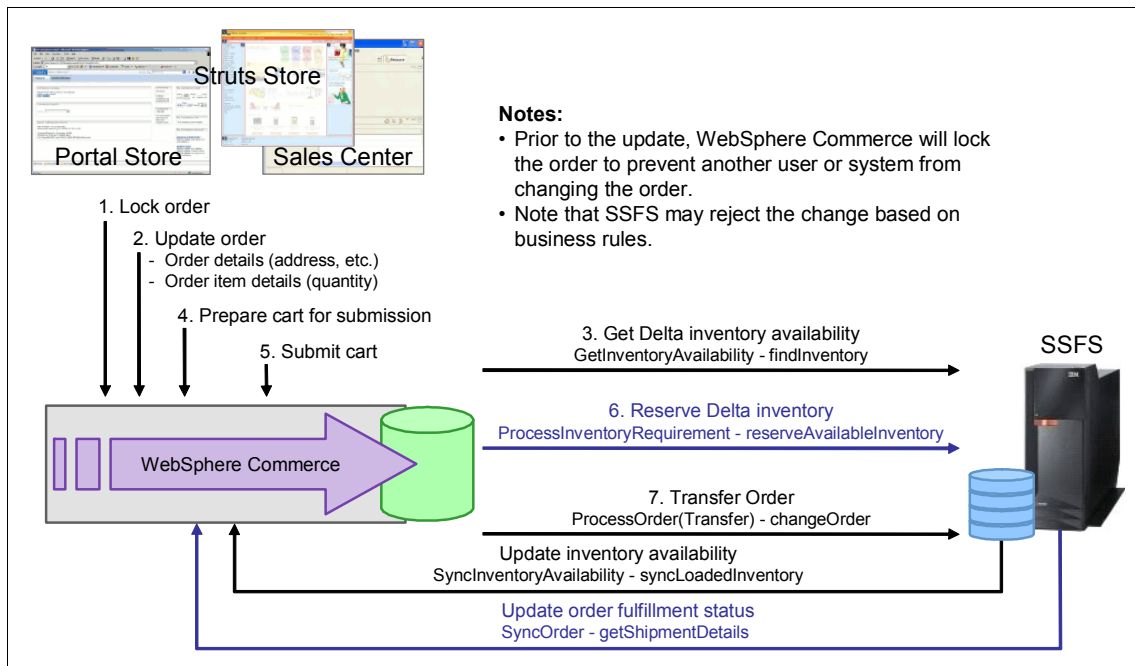


Figure 3-5 Order modification flow

The order modification steps are:

1. Lock the order from within WebSphere Commerce.
2. Update the order.
3. Get inventory availability.
4. Update the cart.
5. Submit the cart.
6. Reserve the inventory in SSFS.
7. Submit the updated order to SSFS with returned status.

3.2.6 Returns processing

An item from an order can be returned if it is considered to be returnable within the business policy. A typical business policy might include the following conditions:

- The purchased item is returnable (for example, not a clearance item).
- The item is shipped.
- The purchase is made within a set number of days.

Figure 3-6 shows the detailed steps for processing returns from within WebSphere Commerce.

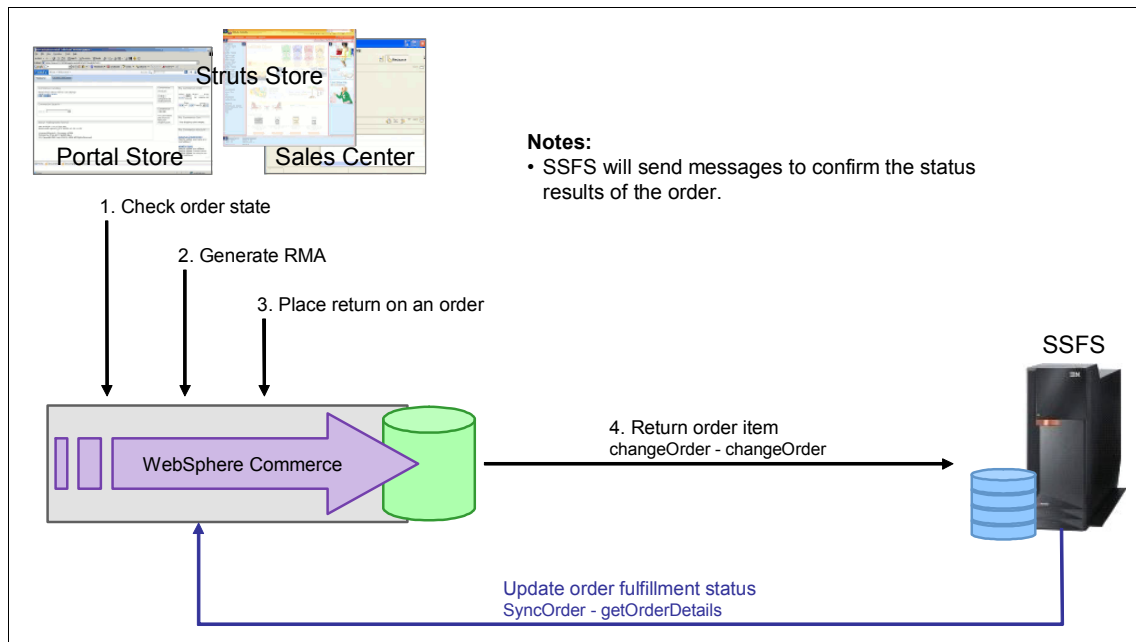


Figure 3-6 Return processing flow

Return processing steps are:

1. Check the order state.
2. Generate a Return materials authorization (RMA).
3. The return is processed in WebSphere Commerce.
4. RMA order information is sent to SSFS.

3.3 Available features in WC

To achieve the SSFS integration for order management, WebSphere Commerce Madison starter store and Madison Mobile starter store JSPs support the following OOB features:

- ▶ Store locator
- ▶ Stock locator
- ▶ Buy-online-pick-up-in-store
- ▶ Buy-online-ship-to-store
- ▶ Reserve online pay in-store
- ▶ DOM integration

3.3.1 Store locator

The store locator feature provides a list of nearby stores based on the input location. The resulting store list can be customized to display a set of stores. Customers can select multiple store locations when checking the availability of a product, however the store locator must be used before the availability of a product can be checked.

3.3.2 Stock locator

The stock locator pulls the availability of a product and shows the availability status on the store page (for example, in stock or out of stock).

3.3.3 Buy-online-pick-up-in-store

This scenario illustrates the concept of using Sterling Commerce DOM to model a store as just another warehouse for storing specific inventory.

3.3.4 Buy-online-ship-to-store

Similar to the buy-online-pick-up-in-store feature, the customer can purchase products online and pick them up at a store location. However, in this scenario, the requested items are currently out-of-stock at the store location. The customer instead chooses to ship and pick up the items at the store location of her choice. Customers can retrieve their purchase at a time of their convenience after it is made available for pickup at their preferred store location. Similar to the buy-online-pick-up-in-store feature, additional sales for the retailer are made possible, as the customer is present in the store when picking up her products.

3.3.5 Reserve online pay in-store

In this case a customer reserves a product online and makes a payment at the store and can pick up their product at the store.

3.3.6 DOM integration

DOM integration enhances the shopping flow in the Madisons starter store, while enhancing back-end system integration with WebSphere Commerce. This integration provides comprehensive coverage of the order life cycle across channels, from capture to fulfillment. After enabling DOM integration, your inventory type is changed to make use of an external inventory system. Therefore, you must implement an external inventory system to use DOM integration functionality in WebSphere Commerce and your storefront.

3.4 Scenarios covered OOB

Below are the three basic scenarios covered OOB in FEP2 release. See the respective chapters to understand more about the each scenario.

- ▶ Chapter 4, “Business scenario: Catalog Browse” on page 57
- ▶ Chapter 5, “Business scenario: Order capture” on page 79
- ▶ Chapter 6, “Business Scenario: Order status” on page 113



Business scenario: Catalog Browse

This chapter explains the catalog browse flows from the user prospective and how to verify those flows. Major functionality of the catalog browsing resides in WebSphere Commerce, and the inventory of the catalog items is managed by Sterling Selling and Fulfillment Suite (SSFS).

This chapter contains the following sections:

- ▶ 4.1, “Scenario introduction” on page 58
- ▶ 4.2, “Prerequisites” on page 58
- ▶ 4.3, “Catalog browse flow” on page 59

4.1 Scenario introduction

WebSphere Commerce provides catalog browse and search functions. The WebSphere Commerce store allow a shopper to browse through the categories and view items or directly use search terms to find a particular item or a list of items.

All of the catalog, categories, and products under categories can be created and managed using WebSphere Commerce Management Center. The inventory part of these items will be configured and managed by SSFS.

WebSphere Commerce will make necessary calls to SSFS to get the inventory for items. WebSphere Commerce shows the *near real-time* inventory. This inventory is based on a locally cached copy of the inventory, which is refreshed periodically from the SSFS Distributed Order Management (DOM).

A DOM integration feature is the ability to show inventory data as stored in a local configurable cache, if the local inventory cache is above a threshold, and do a real-time inventory check only if the local inventory cache is below the configured threshold. This method gives better system performance and also allows a shopper to know immediately whether a product is available to be ordered.

DOM integration can provide sufficient information to help a shopper decide whether to purchase the item online or at a local store. To do so, knowing the inventory at each physical store, as provided by the DOM integration, is the key to the buy-online-and-pickup-in-store (BOPIS) solution.

The out-of-the-box WebSphere Commerce sample online store displays one or multiple selected stores' inventories only after a shopper has decided and selected a physical store from which to pick up the order. This approach gives better scalability to support a large number of physical stores.

4.2 Prerequisites

In the following sections, we point to the prerequisite steps and configuration required for executing the Catalog Browse scenario.

Make sure that you have configured WebSphere Commerce, WebSphere Enterprise Service Bus (WESB), and Sterling Selling and Fulfillment Suite (SSFS) as per Chapter 7, "Installation and configuration" on page 129.

4.3 Catalog browse flow

There are two ways to display inventory during the catalog browse:

- ▶ Real time
- ▶ Cached inventory

Real-time inventory allows shoppers to see the actual inventory of a product as they are browsing the catalog. Having this information available has to be weighed against the performance impact of not fully caching the product pages, which requires reaching out to the inventory system for each shopper for every page. Often, for performance reasons, the real-time request is left for the order submit phase of the shopping scenario rather than the browsing phase.

During the browse phase, it is desirable to cache all information related to the product, including inventory. The real-time request for inventory should be done on an as-needed basis and, if made, should refresh the local cache with the latest view of inventory. Again, for performance reasons, the design should consider a threshold amount of inventory where the real-time inventory is only acquired if the level of inventory falls below a certain threshold, where the threshold is a safe amount of inventory that provides confidence that the inventory has not been fully consumed. Only if the locally cached inventory falls below this threshold is the real-time inventory truly required.

In some cases, for instance with a wine collection, having a threshold amount is insufficient during the browse phase and real-time inventory are desirable. In this case, WebSphere Commerce retrieves the information from different inventory systems and store locations through SSFS DOM interaction if store level inventory is required. To support this, the system must handle the performance impact of the request. WebSphere Commerce supports the concept of physical store object, which contains store information, location information, and location attributes and provides services to retrieve or manipulate physical store data. The store information includes the store name, description, and information describing the store.

The location information includes identifiable information related to a store, such as its geocode, and the geonode of the store. Each physical store also has a cached inventory number, and therefore a shopper can browse and see the physical store specific inventory for better performance.

4.3.1 System interaction diagram

Figure 4-1 shows the scenario flow for catalog browse and core systems involved to complete this flow. There are different possible user interfaces listed in the diagram. For all of these user interfaces, WebSphere Commerce is the first interaction point, and WebSphere Commerce will interact with SSFS to retrieve inventory. Based on this inventory information, the store page will show the availability status for the product.

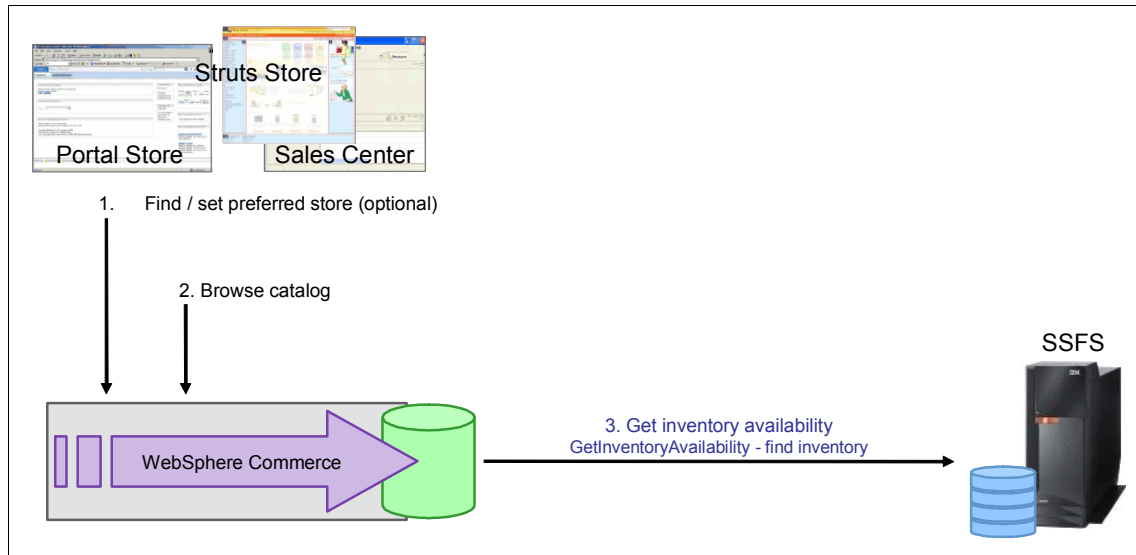


Figure 4-1 Interaction message flow of catalog browse

4.3.2 Flow diagram

Figure 4-2 shows possible flows for catalog browse. This involves finding an item through different search options available in the store, and top-down catalog browse from the home page to the product display page. If the fast finder option is enabled for the store, you can find items using the fast finder.

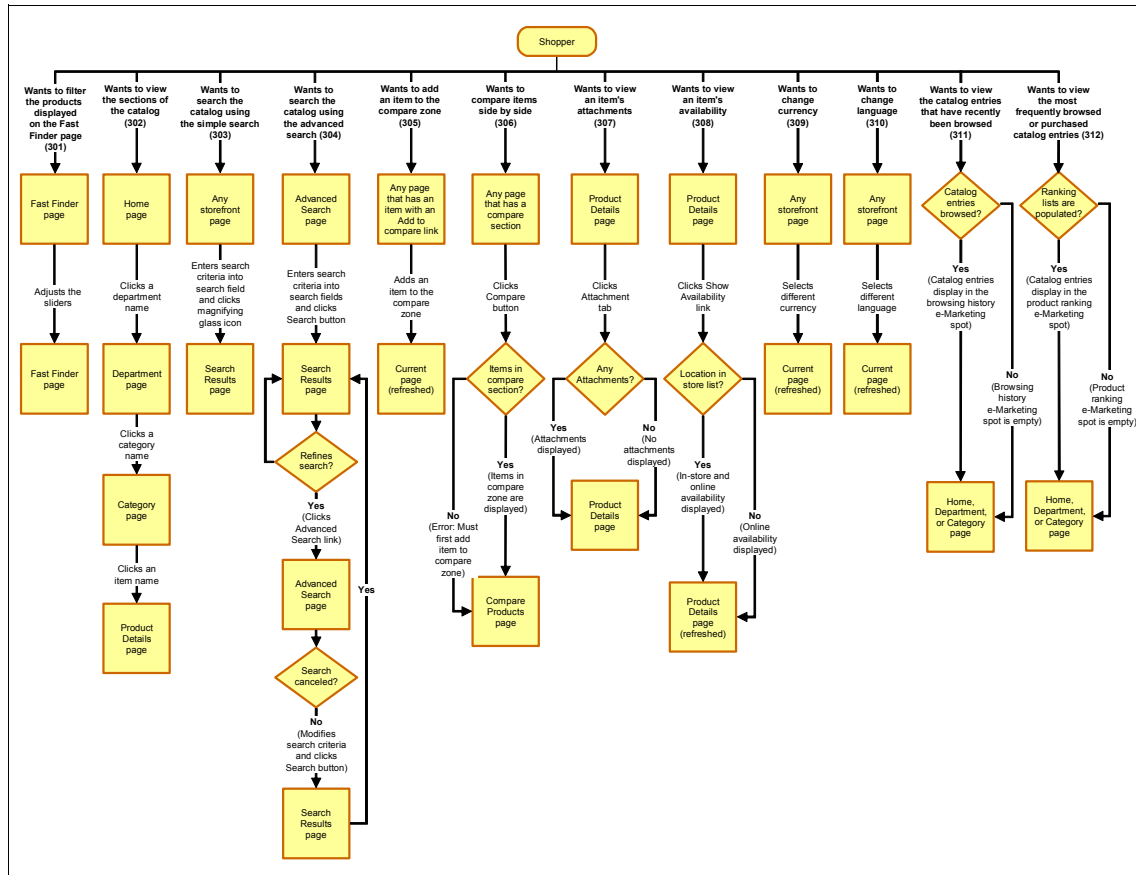


Figure 4-2 Catalog browse flow diagram

4.3.3 Actions and subsystems in the scenario

Table 4-1 is a summary of the scenario and the different systems that are involved to complete this scenario.

Table 4-1 Scenario summary information for catalog browse

Overview	A shopper accesses the store as a guest or register user. Browse the catalog using page navigation, or search for an item using the search option in the storefront. In the product display page, the shopper checks for item inventory.
Channel	Online storefront.
Actor/system	Shopper.
Subsystem	WebSphere Commerce, SSFS.
Trigger event	Shopper wants to find and check item availability.
Additional information	All the product information comes from WebSphere Commerce. Inventory comes from SSFS.

4.3.4 Execution flow

This section provides detailed steps for the catalog browsing and how to navigate to an item to check the inventory. When a user reaches the product display page, WebSphere Commerce makes an external call to pull the inventory information from SSFS.

Note: For testing purposes in the lab, inventory caching is configured to expire quickly. In production, caching can be configured to have a longer time before expiration, based on constraints of system resources, availability of back-end DOM, and so on.

The steps are:

1. Launch the Madisons store using the following store URL:

`http://<hostname>/webapp/wcs/stores/servlet/TopCategories_<storeId>_<catalogid>`

The Madisons store URI will navigate to the store home page (Figure 4-3).



Figure 4-3 Online Madison store

2. Click **Furniture**, which takes you to subcategory page (Figure 4-4).
3. Click **Lounge Chair** from the subcategory page (Figure 4-4).

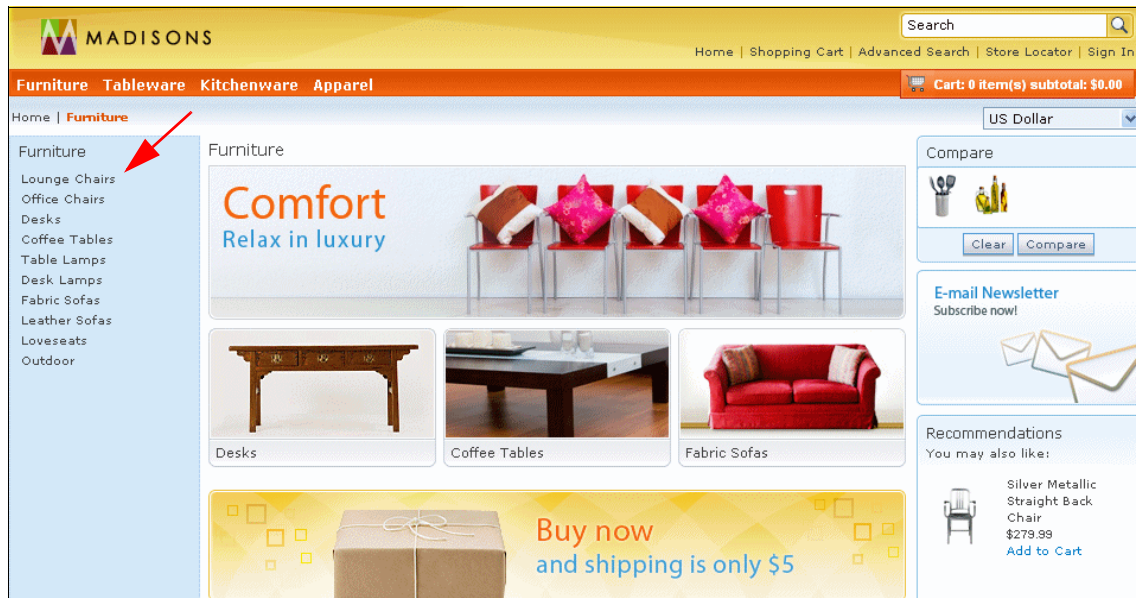


Figure 4-4 Store subcategories page

4. You now see the product list page (Figure 4-5) showing all the products under the Lounge Chair subcategory. Click **White Fabric Roll Arm Chair** on the product list page.

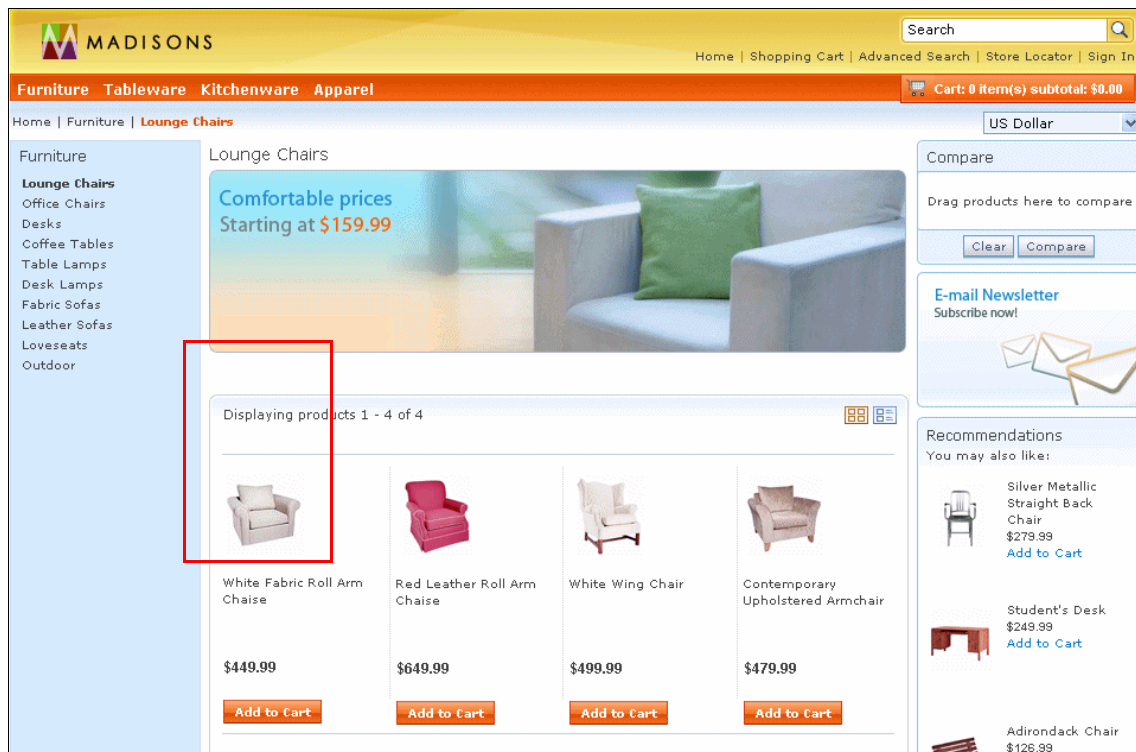


Figure 4-5 List of products available in the category

Now you are at the product display page for “White Fabric Roll Arm Chair” (Figure 4-7 on page 67).

As an alternative to the previous way to get product information, perform step on page 63 again. This time you get the list of products using the search option available on the right side at the top of the store page. Next:

1. Enter **White Fabric Roll Arm Chair** in the search box and click the search icon (the magnifying glass). This takes you to the product search result page (Figure 4-6).

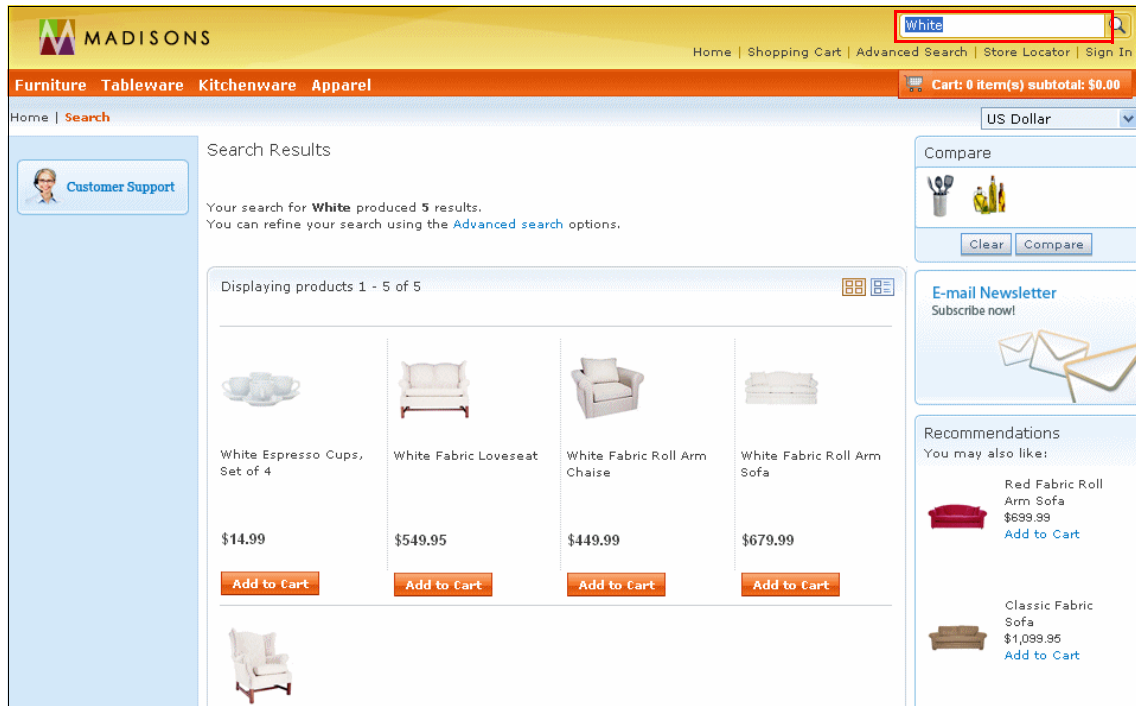


Figure 4-6 Product search result page

2. Click **White Fabric Roll Arm Chair** from the product list page, which takes you to the product display page for “White Fabric Roll Arm Chair” (Figure 4-7 on page 67).

On the product details page you will find all the product information and inventory of the product. Basically, this shows the availability status. WebSphere Commerce fetches this information from the SSFS inventory management system. As discussed in 4.1, “Scenario introduction” on page 58, it gets information from WebSphere Commerce cache, which updates itself as per the threshold values settings. If the product is available online, you will see that it is in stock under Online Availability (Figure 4-7).

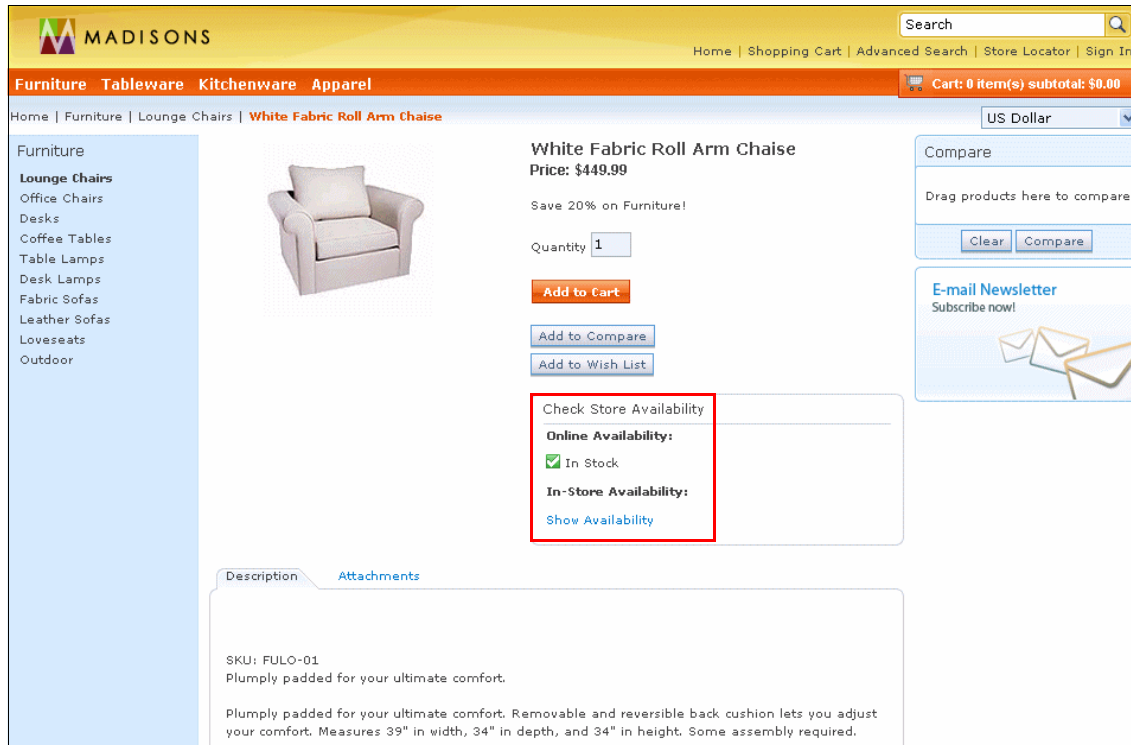


Figure 4-7 Item stock availability

4.3.5 Alternative buy-online-and-pickup-in-store (BOPIS) flow

This scenario illustrates the concept of using SSFS DOM to model a store as just another warehouse for store specific inventory.

Here the customer can purchase products online and pick them up at a store. However, in this scenario, the requested items are currently out of stock at the online store location. The customer can choose the nearest store by giving the location where he wants to find a store. Customers can select multiple store locations when checking the availability of a product. However, the store locator must be used before the availability of a product can be checked.

1. Follow step on page 63-4 on page 65 in 4.3.4, “Execution flow” on page 62, to reach the product display page for “White Fabric Roll Arm Chair” (Figure 4-7 on page 67).

Figure 4-7 on page 67 shows the page that you will see if the item is not in stock for the online store. In the BOPIS scenario, you can select a store from which you want to pick up the item once the order is submitted online.

2. Click **Select Store** on the product display page (Figure 4-8).

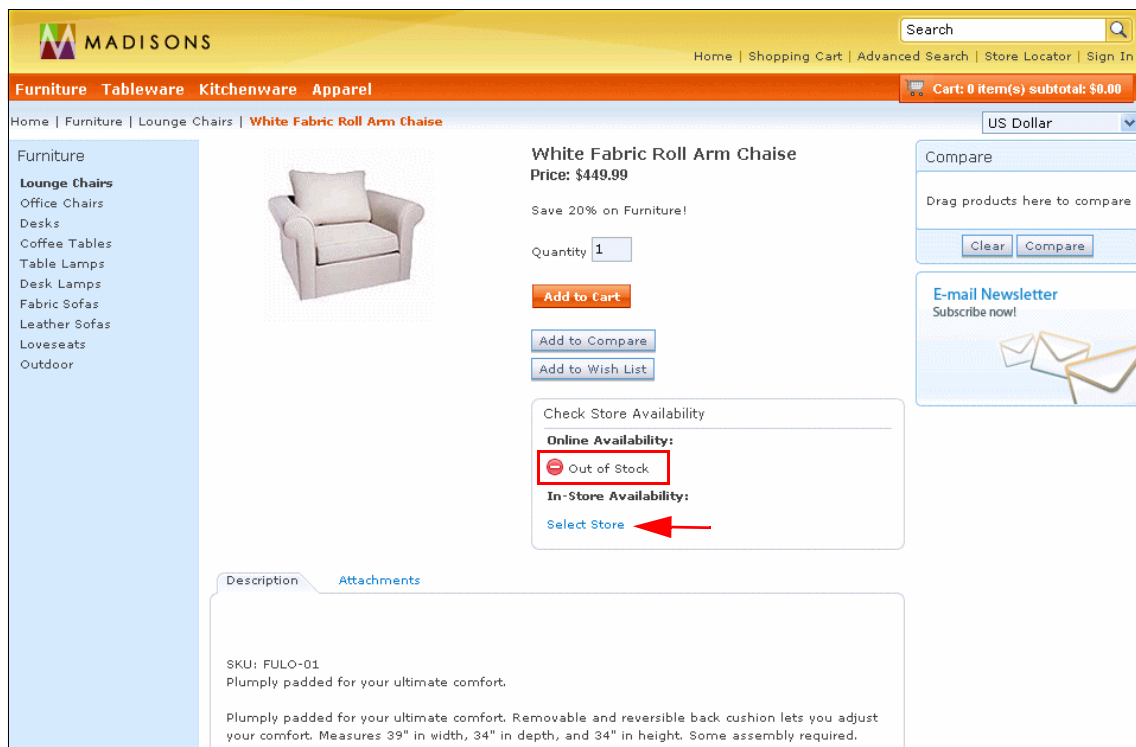


Figure 4-8 Item out of stock flow

3. Navigate to the store locator page (Figure 4-9), where you can select a store nearby from which to pick up the item later. Select your country (Canada, in this case), state (Ontario), and city (Toronto) and click **GO**.


MADISONS Search

Home | Shopping Cart | Advanced Search | Store Locator | Sign In

Furniture Tableware Kitchenware Apparel Cart: 0 item(s) subtotal: \$0.00

Home | Furniture | Lounge Chairs | White Fabric Roll Arm Chaise | [Check Availability](#) US Dollar

Check availability at a store near you

 [White Fabric Roll Arm Chaise](#)
Price: \$449.99

Your Store List [Hide](#)

Your store list does not contain any store locations. Use the Store Locator to add store locations.

Store Locator [Select a location to find a store near you.](#)

Country: [Canada](#) State/Province: [Ontario](#) City: [Toronto](#) **GO**

Store Locator Results [Select one or more stores to check product availability.](#)

Select your search criteria and click GO to find stores nearest to your location.
Add a store location to your store list.
[Continue Shopping](#)

Figure 4-9 Select a location to get a list of stores

4. You will see a list of stores near the location entered (Figure 4-10). Click **Add to store list** for “Bathurst Plaza.”

The screenshot shows the MADISONS website interface. At the top, there's a navigation bar with the MADISONS logo, a search bar, and links for Home, Shopping Cart, Advanced Search, Store Locator, and Sign In. Below this is a secondary navigation bar with categories: Furniture, Tableware, Kitchenware, and Apparel. A cart summary shows 0 items for \$0.00. The main content area is titled 'Check availability at a store near you' and features a product listing for a 'White Fabric Roll Arm Chaise' priced at \$449.99. Below the product listing is a section for 'Your Store List' which currently is empty, with a message: 'Your store list does not contain any store locations. Use the Store Locator to add store locations.' The 'Store Locator' section allows users to select a location to find a store near them, with dropdown menus for Country (Canada), State/Province (Ontario), and City (Toronto), and a 'GO' button. Below the search results, there's a table of 'Store Locator Results' for the selected area. The table has three columns: 'STORE NAME AND ADDRESS', 'HOURS', and 'SELECT STORE'. Two stores are listed: 'Alton Towers Plaza' and 'Bathurst Plaza'. The 'Bathurst Plaza' entry is highlighted with a red box, and its 'Add to store list' button is also highlighted.

STORE NAME AND ADDRESS	HOURS	SELECT STORE
Alton Towers Plaza 250 Alton Towers Circle Toronto, Ontario M1V 3Z4 416.326.8647	Mon-Fri: 10am - 9pm Sat: 9am - 7pm Sun: 11am - 6pm	Add to store list
Bathurst Plaza 781 Bathurst St.	Mon-Fri: 10am - 9pm Sat: 9am - 7pm	Add to store list

Figure 4-10 List of stores in selected area

You see a list of stores added for checking inventory availability (Figure 4-11).

Your Store List

Hide

STORE NAME AND ADDRESS	HOURS	
Alton Towers Plaza 250 Alton Towers Circle Toronto, Ontario M1V 3Z4 416.326.8647	Mon-Fri: 10am - 9pm Sat: 9am - 7pm Sun: 11am - 6pm	Remove
Bathurst Plaza 781 Bathurst St Toronto, Ontario M6G 1B4 416.768.3265	Mon-Fri: 10am - 9pm Sat: 9am - 7pm Sun: 11am - 6pm	Remove
Bay Plaza 180 Bay ST Toronto, Ontario M5J 2W3 416.287.6451	Mon-Fri: 10am - 9pm Sat: 9am - 7pm Sun: 11am - 6pm	Remove

Store Locator

Select a location to find a store near you.

Country:

State/Province:

City:

Canada

Ontario

Toronto

GO

Store Locator Results

Select one or more stores to check product availability.

STORE NAME AND ADDRESS	HOURS	SELECT STORE
Alton Towers Plaza 250 Alton Towers Circle Toronto, Ontario M1V 3Z4 416.326.8647	Mon-Fri: 10am - 9pm Sat: 9am - 7pm Sun: 11am - 6pm	Add to store list
Bathurst Plaza 781 Bathurst St Toronto, Ontario M6G 1B4 416.768.3265	Mon-Fri: 10am - 9pm Sat: 9am - 7pm Sun: 11am - 6pm	Add to store list

Figure 4-11 Selection of store to check inventory

5. Click **Continue shopping** (Figure 4-12).

Sherway Gardens
25 West Mall
Etobicoke, Ontario M9B 5Z9
905.456.7658

Mon-Fri: 10am - 9pm
Sat: 9am - 7pm
Sun: 11am - 6pm

[Add to store list](#)

University Avenue
222 University Ave
Toronto, Ontario M5H 4B8
416.222.3333

Mon-Fri: 10am - 9pm
Sat: 9am - 7pm
Sun: 11am - 6pm

[Add to store list](#)

Yorkdale Centre
3401 Dufferin Street
North York, Ontario M2N 1A1
905.411.1278

Mon-Fri: 10am - 9pm
Sat: 9am - 7pm
Sun: 11am - 6pm

[Add to store list](#)

Add a store location to your store list.

[Continue Shopping](#)

Figure 4-12 Returning to main window by clicking Continue Shopping

6. You are taken back to the product display page to proceed to shopping information. Click **In Store availability** on the product display page to see the availability of the inventory for each store (Figure 4-13).

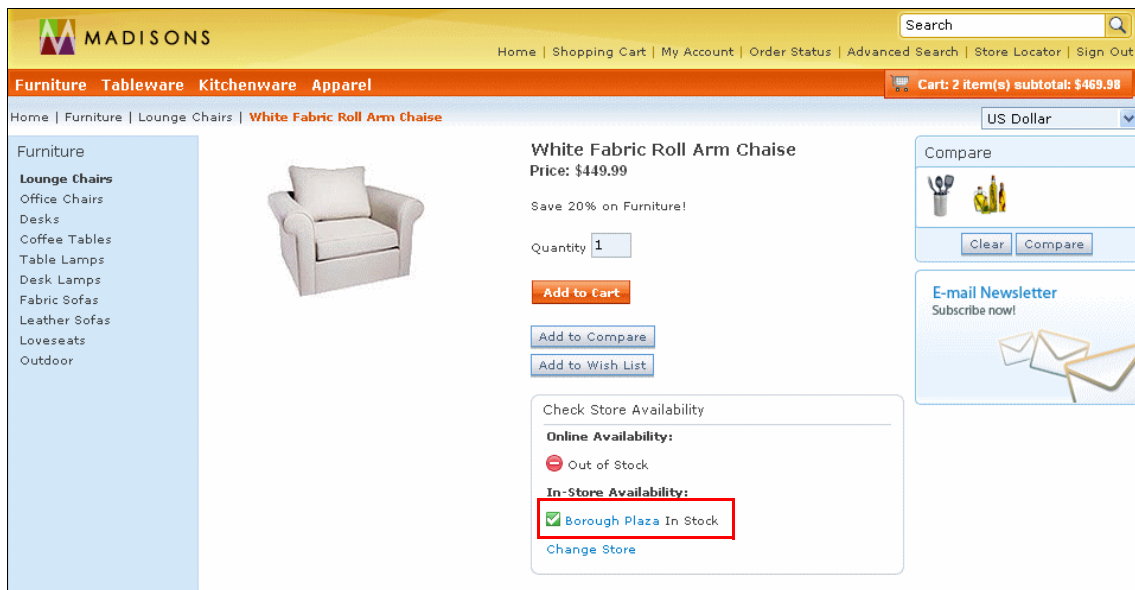


Figure 4-13 Availability of stock in store


4.3.6 Exception flow verification

This scenario helps verify how a storefront will be effected by changing inventory in SSFS.

1. Log in to the SSFS Application console using the following URL:
`http://<host name>/smcfs/console/login.jsp`
2. Navigate to Item inventory page by clicking **Inventory** → **Inventory Console**.

3. Find inventory for a specific item using the item SKU FUL0-0101.

The display (Figure 4-14) shows the inventory, where the item quantity is set to 100 in all the stores.

 Supply Type Details

Help

Close

Item Information

Item ID FUL0-0101

Product Class

Unit Of Measure EACH

Description

Supply Type Onhand

Supply Information

Node	ETA	PO #	Segment Type	Segment	Quantity
Bay Plaza					100.00
Borough Plaza					100.00
DANFORTH Plaza					100.00
Eaton Centre					100.00
Keele Plaza					100.00
Madisons					100.00
Sherway Gardens					100.00
University Avenue					100.00
Yorkdale Centre					100.00

Figure 4-14 Inventory details in SSFS

- Follow steps 1 - 4 in 4.3.4, “Execution flow” on page 62, to reach the product display page for “White Fabric Roll Arm Chair” that shows the availability status as in stock for all the stores (Figure 4-15).

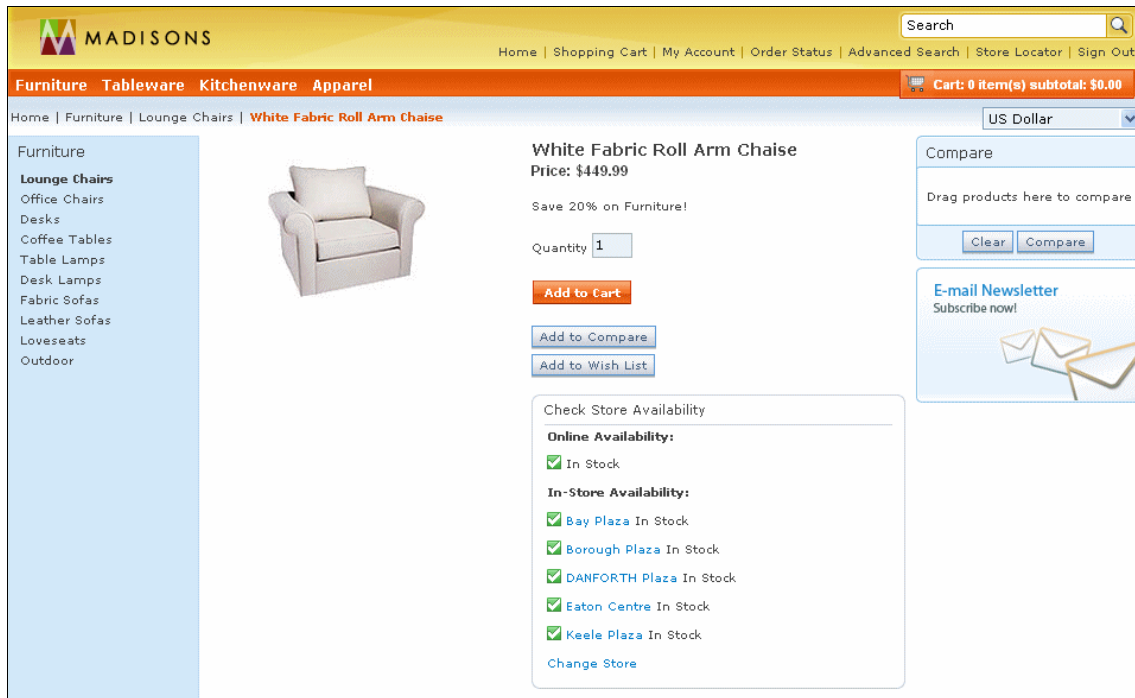


Figure 4-15 Storefront inventory availability

5. Follow step 1 on page 72 through step 3 on page 73. Go to the application console and decrease the inventory for the online store so that it becomes 0 (Figure 4-16).

Application Console Administrator

Alerts Order Inventory Supply Logistics Reverse Logistics Business Intelligence Configuration System Help

Adjust Inventory Save Help

Item Details

Item ID: FULO-0101 Product Class Unit Of Measure: EACH

Inventory Information

Ship Node: Madisons Supply Type: Onhand Segment Type: Segment:

Adjustments

Availability: ☒ Track ☐ Infinite

Quantity: 100.00

Decrease By: 100

References

Reference #1
Reference #2
Reference #3
Reference #4
Reference #5

Modification Reason

Reason Code:
Reason Text:

Figure 4-16 Set the inventory to 0 for the online store

6. Now reload the product display page for “White Fabric Roll Arm Chair.”

The display shows the item as out of stock for the online store but in stock in store (Figure 4-17).

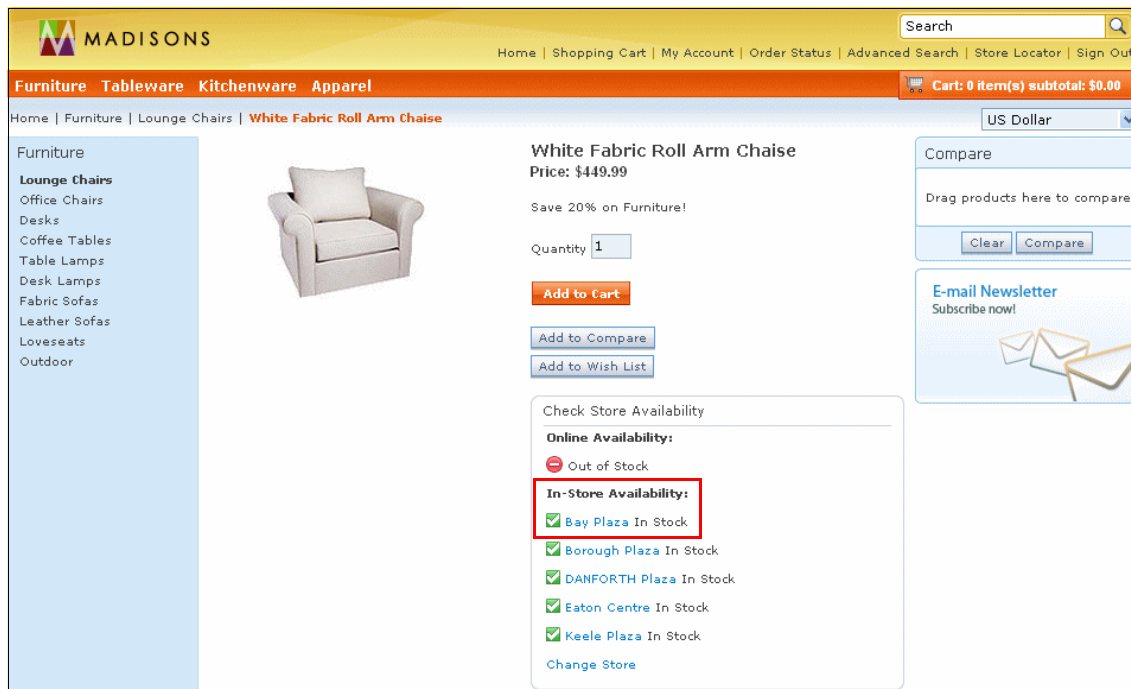


Figure 4-17 Online store item availability out of stock in storefront

7. Follow step 1 on page 72 through step 3 on page 73 again to get to the application console, and decrease the inventory for the item for all the stores so that the inventory becomes 0 (Figure 4-18).

Application Console Administrator

Alerts Order Inventory Supply Logistics Reverse Logistics Business Intelligence Configuration System Help

Adjust Inventory Save Help

Item Details

Item ID: FULO-0101 Product Class Unit Of Measure: EACH

Inventory Information

Ship Node: [Dropdown] Supply Type: Onhand Segment Type: [Dropdown] Segment: [Input]

Adjustment

Available: Infinite

Quantity: 0

Decrease By: [Button]

References

Reference #1 [Input]
Reference #2 [Input]
Reference #3 [Input]
Reference #4 [Input]
Reference #5 [Input]

Modification Reason

Reason Code: [Dropdown]
Reason Text: [Input]

Figure 4-18 Decrease inventory for all of the stores

8. Now reload the product display page for “White Fabric Roll Arm Chair.” The page shows that the item is out of stock for all the stores.

In this chapter, we discussed how the catalog browse works in WebSphere Commerce and during this operation how it interacts with SSFS via a enterprise service bus infrastructure provided by WebSphere Enterprise Service Bus.



Business scenario: Order capture

This chapter discusses the order capture scenario, which includes adding an item to the cart and checkout processes. It includes step-by-step instructions for shoppers to place an order via the web. It also provides details about system integration and flow diagrams and discusses results in buy-online-pay-in-store scenarios.

After catalog browsing, once a shopper is ready with his shopping cart, the shopper can check out. This is where the actual inventory of the items in the shopping cart is locked, and the order information is packaged by WebSphere Commerce and sent to the Sterling Selling and Fulfillment Suite (SSFS), which fulfills the order and provides the status throughout the order life cycle.

This chapter contains the following sections:

- ▶ 5.1, “Scenario introduction” on page 80
- ▶ 5.2, “Prerequisites” on page 82
- ▶ 5.3, “Adding item to cart” on page 82
- ▶ 5.4, “Checkout flow” on page 91
- ▶ 5.5, “Buy-online-pickup-in-store (BOPIS) scenario” on page 105

5.1 Scenario introduction

This section discusses the order capture scenario. In this scenario, a shopper selects items to buy from the online storefront. Once that is done, the order information is packaged by WebSphere Commerce and sent to the SSFS system, which fulfills the order and provides the status throughout the order life cycle.

When the shopper selects the Add to Cart option, WebSphere Commerce checks for the inventory availability and adds the item to the cart if the item is available. After items are added to the shopping cart in WebSphere Commerce, a message is sent to the SSFS DOM to query the real-time inventory availability. If the inventory is below the threshold, a real-time inventory check at this step of the order capture process is required by the site's business process. However, the best practice is to check inventory only at order submit time for system performance considerations. A physical store inventory can also be handled in the same way if the SSFS DOM system supports inventory by the store. SSFS can be configured for this. After all items have been added to the shopping cart, the order is then prepared for submission. At this point, taxes, shipping charges, and any promotional discounts are calculated and applied to the order. A message is sent to the Sterling Commerce DOM to reserve the inventory for the items in the order. The order is then submitted and the order details are transferred to SSFS using the transfer order message. The order is now under the control of SSFS.

Fraud check is usually executed together with payment authorization after the payment information has been entered and before submitting the order to DOM. However, fraud check, payment authorization, payment capture, and subsequent re-execution of these steps due to order modification are out of scope of this publication.

Figure 5-1 shows the order capture flow across user interface, WebSphere Commerce, and SSFS. This flow mainly involves six transactions:

- ▶ The WebSphere Commerce system handles the following transactions:
 - Add item to shopping cart.
 - Prepare cart for submission.
 - Submit cart.
- ▶ SSFS handles the following transactions sent by WebSphere Commerce:
 - Check inventory availability.
 - Reserve inventory.
 - Transfer order for process.

Figure 5-1 shows the transactions in the order flow.

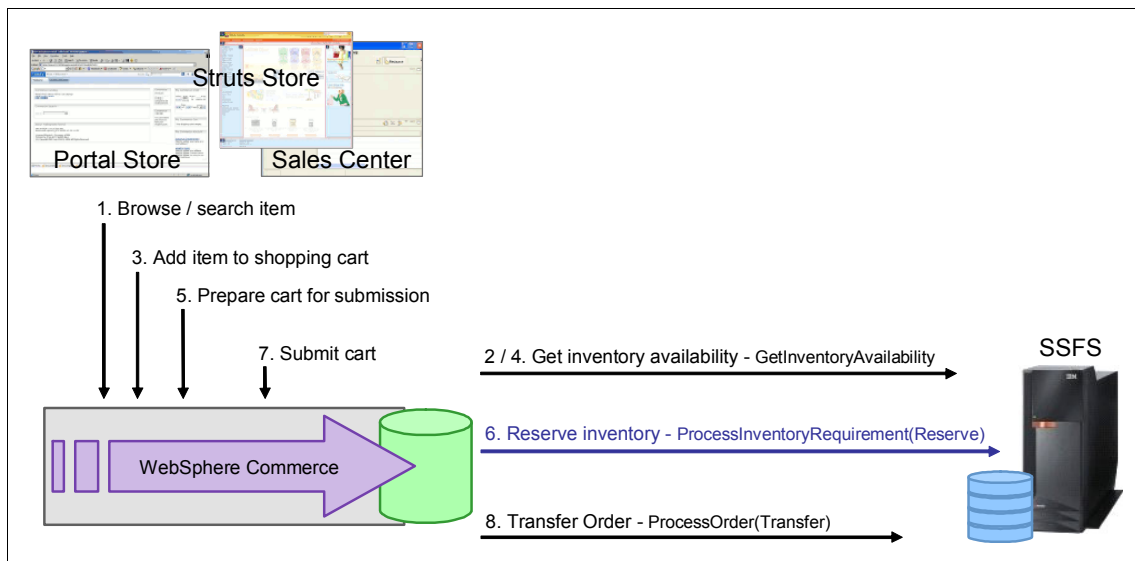


Figure 5-1 Order capture message flow

The following sections discuss two subscenarios that cover the order capture scenario:

- ▶ Add item to cart.
- ▶ Check out.

5.2 Prerequisites

This section discusses the prerequisite steps and configuration required for executing the order capture scenario.

Ensure that you have configured WebSphere Commerce, WebSphere Enterprise Service Bus (WESB), and Sterling Selling and Fulfillment Suite (SSFS) as per Chapter 7, “Installation and configuration” on page 129.

5.3 Adding item to cart

This section covers the subscenario in which the user adds items to the shopping cart and proceeds to the checkout. The operation for checkout is discussed later in this chapter.

This section discusses the architecture flow of the add-to-cart transaction. This section includes the steps leading up to the checkout page. It shows how the front store interacts with DOM/OMS through WebSphere Commerce. In our case, this is SSFS. Figure 5-2 shows a architectural flow for adding an item to a cart.

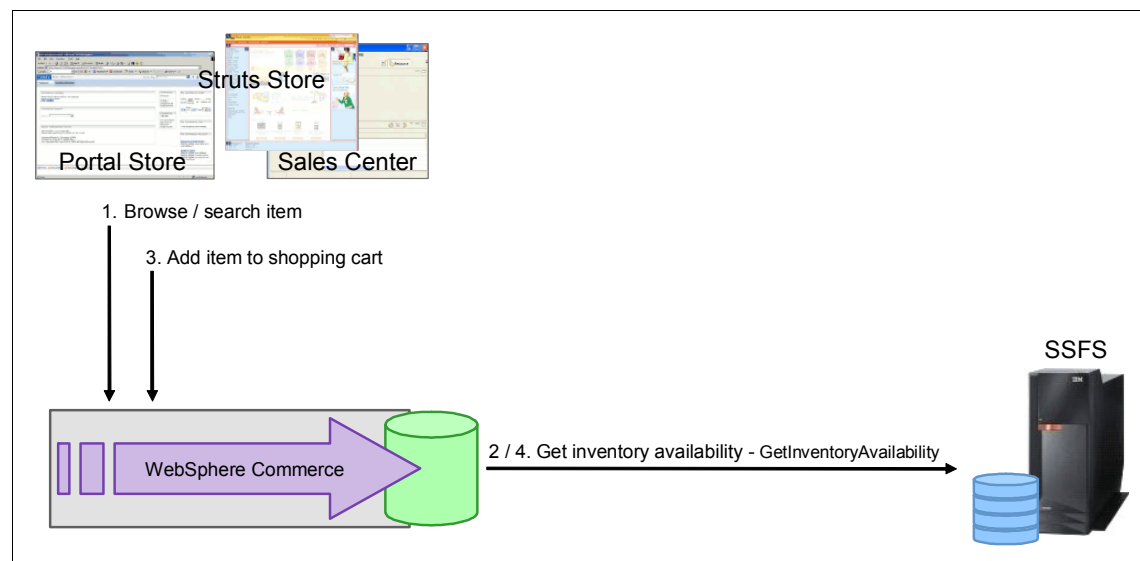


Figure 5-2 Add item to cart flow

If the item is available in cache, it does not check with SSFS for inventory availability. If the items added are above the threshold value, it goes back to SSFS and asks for real-time inventory.

5.3.1 Flow diagram of adding item to a cart

This section discusses the add-to-cart flow, including alternate ways to interact with the front-end store provided by WebSphere Commerce.

Figure 5-3 shows the flow diagram. It provides a detailed overview of use case flows.

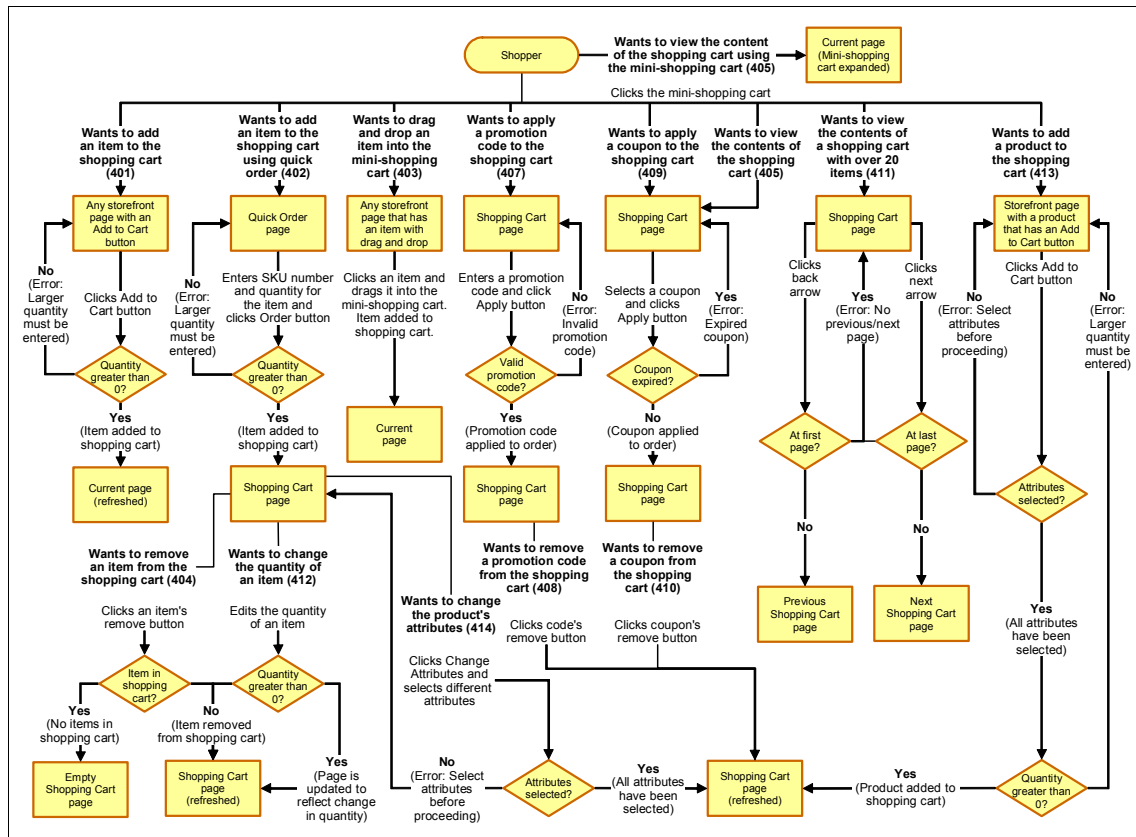


Figure 5-3 Flow diagram for adding an item to the cart

5.3.2 Actions and subsystems in the scenario

This section provides an overview of actions and events triggered when a user follows the add-to-cart flow. In our case, the channel is an online storefront. We use Madison Store for the testing scenario. WebSphere Commerce handles the storefront operation, interacting with SSFS to check inventory availability.

Table 5-1 provides a summary of the actions and systems involved in this flow. It also provides a link for further details.

Table 5-1 Summary of actions and systems involved

Overview	A shopper accesses the store and adds an item to the shopping cart.
Channel	Online storefront.
Actor/system	Shopper, WebSphere Commerce.
Subsystem	SSFS.
Trigger event	Shopper wants to add an item to the shopping cart.
Additional information	http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.madisons-starterstore.doc/refs/rsmmadisonpshoppingcart.htm

5.3.3 Scenario overview and system impact results

This section discusses the action and response (Table 5-2) of the add-to-cart flow. A shopper adds an item to the shopping cart, and inventory is reserved in SSFS after the order is placed when the shopper clicks Order.

Table 5-2 Action/response of scenario

Action	Response
The shopper optionally selects a fulfillment option, then clicks a link to add the item to his shopping cart.	The item is added to the shopping cart, and inventory is reserved in SSFS after the order is placed when the customer clicks Order.

The shopper interaction impacts the inventory level at WebSphere Commerce and SSFS. WebSphere Commerce keeps the inventory level in cache, causing changes on the page based on the inventory availability in the cache. In case of inventory requested going beyond the threshold value, it checks the inventory level with SSFS and updates the cache accordingly. If the cache is not involved, a real-time call is made to SSFS. See Chapter 8, “Integration implementation” on page 195, for more information.

5.3.4 Execution flow

Note: For testing purposes in the lab, inventory caching is configured to expire quickly. In production, caching can be configured to have a longer time before expiration, based on constraints of system resources, availability of back-end DOM, and so on.

The steps to the add-item-to-cart flow, including the exception flows, help you understand the shoppers' view while interacting with WebSphere Commerce (integrated with SSFS). After the item is decided upon, the shopper clicks Add to Cart. Perform the following steps:

1. Select **White Fabric Roll Arm Chaise**. Enter 1 in the Quantity field. Click **Add to Cart**. This event checks the inventory in the WebSphere Commerce cache and adds the item to the cart. Figure 5-4 shows the initial page where the shopper clicks Add to Cart.

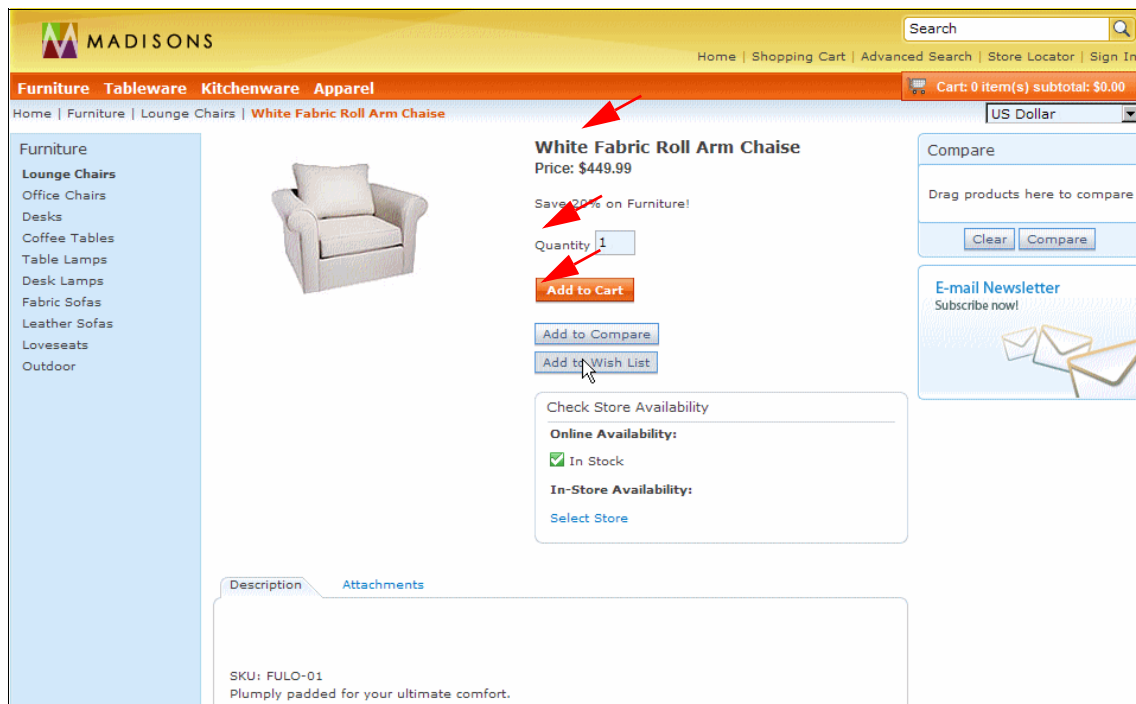


Figure 5-4 Add to cart window

Figure 5-5 shows a cart containing the selected item and any freebie associated with it. You can see that it shows the item as in stock. This information is fetched from SSFS. If caching is enabled, it first hits the WebSphere Commerce cache to get this information. After the shopper clicks Add to Cart, you can see that the item is added into the cart.

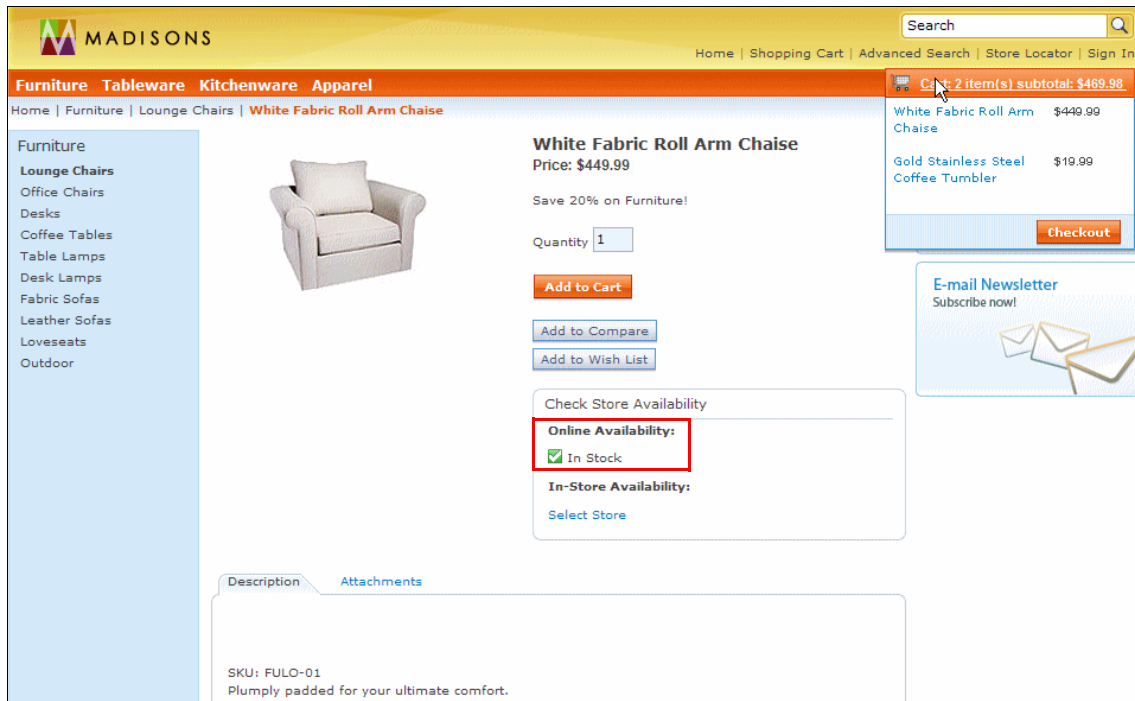


Figure 5-5 Item added to the cart

The shopper also sees a confirmation message, and the cart shows that the items have been added. Figure 5-6 shows a message presented after the item is added to the cart.

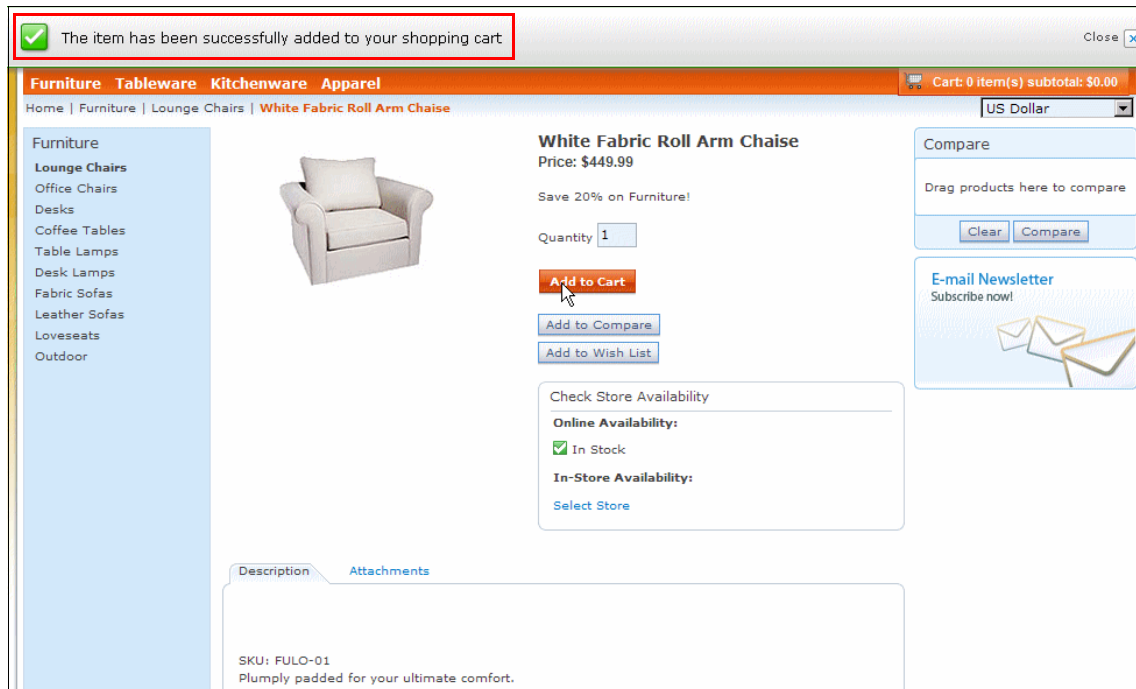




Figure 5-6 Item successfully added message

2. Click **Cart 2 item(s)** on top right of the page. Click in the Quantity text box and change the quantity to 3. Shoppers can decide to change any attribute of item selected. After processing starts, it updates the order information accordingly via an AJAX call. Figure 5-7 shows a page where the shopper changes the attribute value for an item. In this case, it is a quantity of an item.

The screenshot shows the Madisons Shopping Cart page. The header includes the Madisons logo, navigation links (Home, Shopping Cart, Advanced Search, Store Locator, Sign In), and a search bar. The cart summary at the top right indicates 'Cart: 2 item(s) subtotal: \$469.98'. The main cart area lists two items:

PRODUCT	AVAILABILITY	QTY	EACH	TOTAL
 White Fabric Roll Arm Chaise SKU: FULO-0101 Remove	In-Stock	1	\$449.99	\$449.99
Save 20% on Furniture!				(\$90.00)
 Gold Stainless Steel Coffee Tumbler SKU: AC_1701 Free Gift	In-Stock	1	\$19.99	Free

Below the items, there is a promotional code field with an 'Apply' button. The order summary on the right shows: Order Subtotal: \$469.98, Product Discounts: (\$90.00), Discount: (\$19.99), and Order Total: \$359.99. On the far right, there are recommendations for other products like 'Olive Oil Gift Set' and '5-Piece Kitchen Utensil Set'.

Figure 5-7 Cart attribute change window

Figure 5-8 shows a page where, after you change focus out of the attribute value text box, an AJAX request is generated to update the cart. Once done, you can see that the respected values are changed.

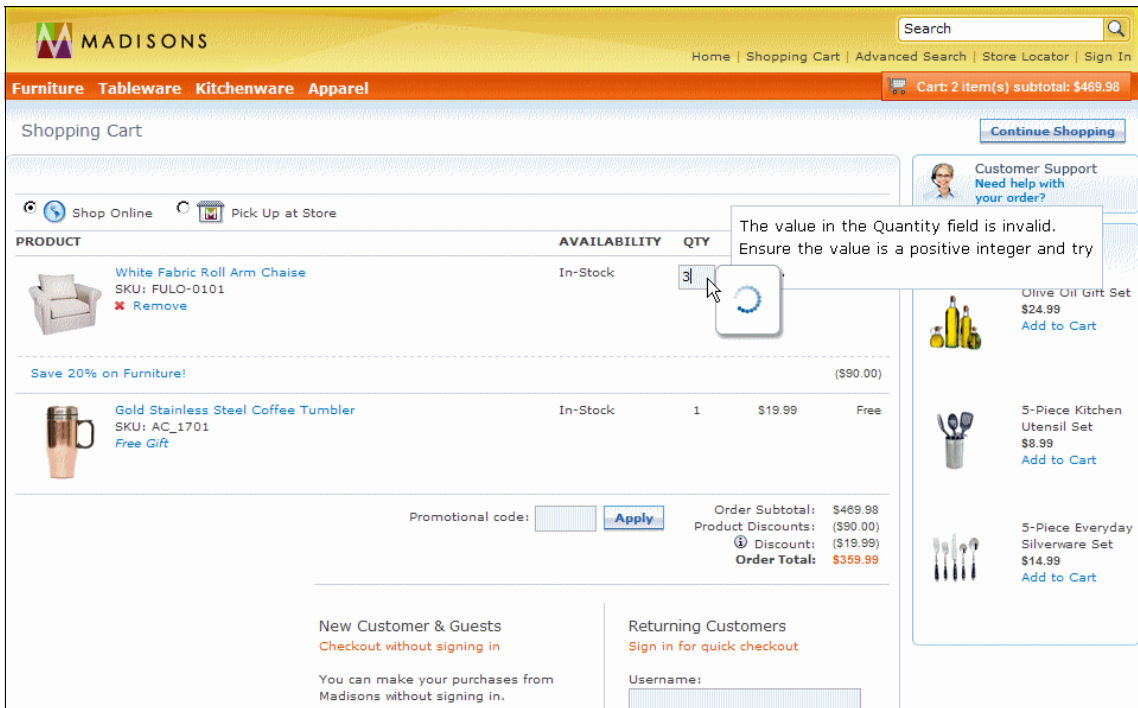


Figure 5-8 Update item attribute in cart

Figure 5-9 shows a page with updated attributes and corresponding changes in the cart.

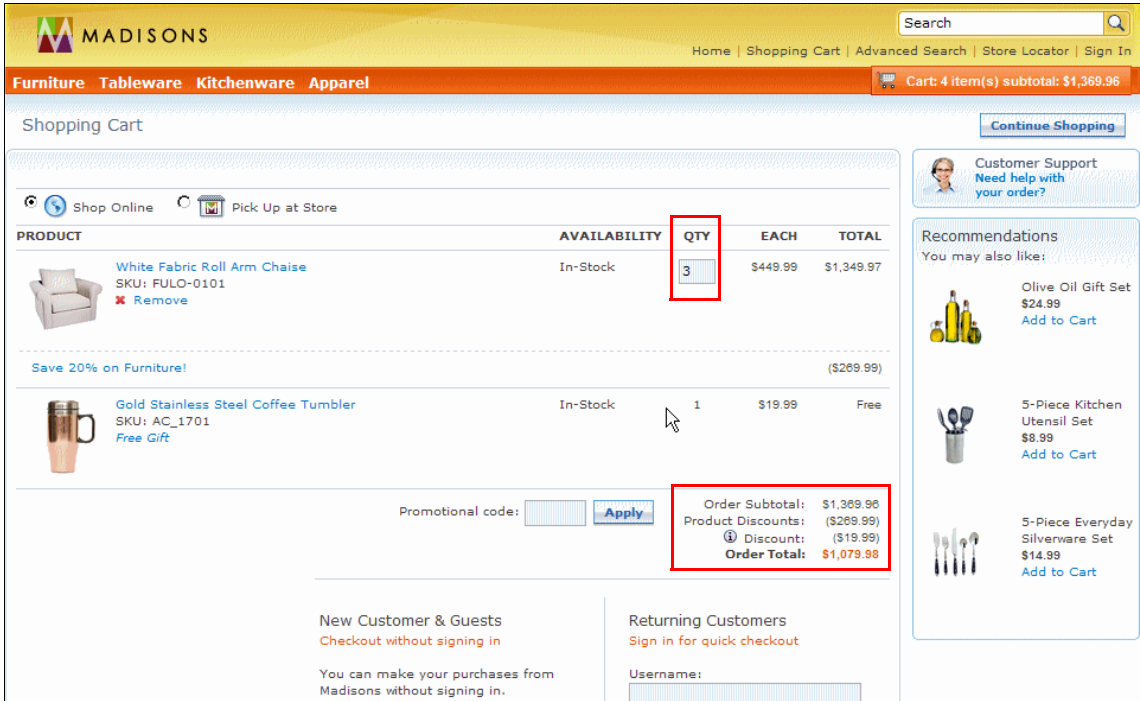


Figure 5-9 Shopping cart with changed quantity and order details

At this point, shopper can review the information and decide to check out.

5.3.5 Exception flows while entering item to a cart

This section discusses exception flows when a shopper interacts with the online storefront. These events also generate a request to SSFS to verify the inventory availability (Table 5-3).

Table 5-3 Exception flow at integration level

Action	Response
The shopper attempts to increase the quantity of an item, but doing so would cause the threshold value for the maximum number of items to be surpassed.	<ol style="list-style-type: none">1. Quantity is not changed.2. Error message is displayed that informs the shopper that if the items were to be added, the shopping cart's threshold would be exceeded.

Refer to Chapter 8, “Integration implementation” on page 195, for more information.

5.4 Checkout flow

This section discusses the checkout process subscenario of the order capture scenario. During this process, you provide the relevant information to finish the transaction, and it reserves the inventory in SSFS.

5.4.1 System integration diagram

This section discusses the architecture flow of the checkout transaction, including the steps for the order confirmation. It shows how the store interacts with DOM/OMS through WebSphere Commerce using SSFS.

Figure 5-10 shows an architecture diagram of the checkout flow. After you submit the order, it is captured by WebSphere Commerce, which passes it to SSFS to process further.

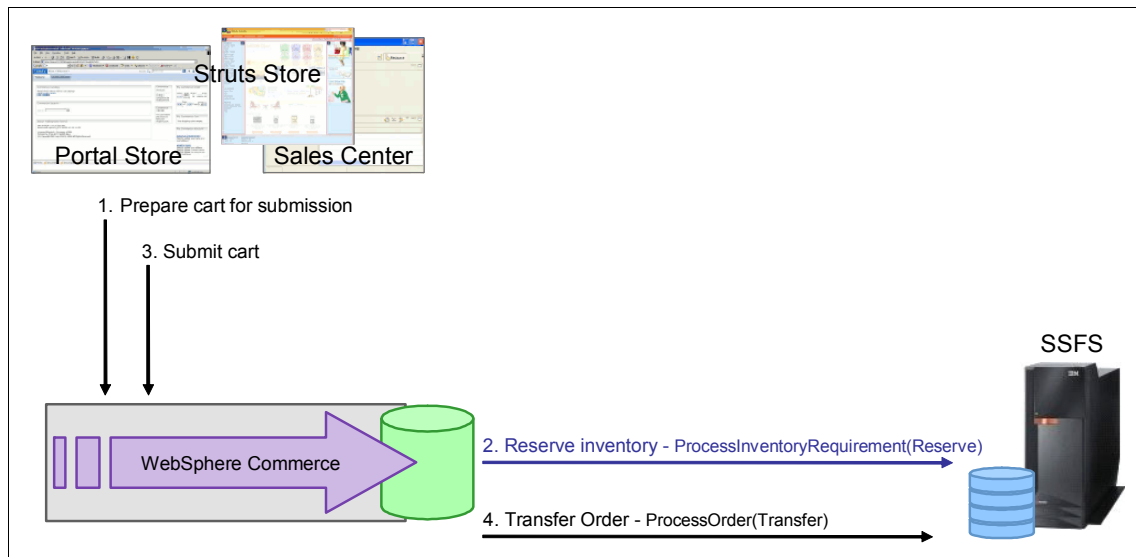


Figure 5-10 Check out flow

Note: There could be a delay in passing an order from WebSphere Commerce to SSFS due to this latency. However, the order might not reflect that immediately in SSFS.

5.4.2 Flow diagram

This section discusses the checkout flow, including alternate ways to interact with the frontend store after you reach the checkout page provided by the WebSphere Commerce online storefront module. Figure 5-11 shows a flow diagram with all possible flows for a checkout process. Figure 5-11 contains a single flow showing the order placement process.

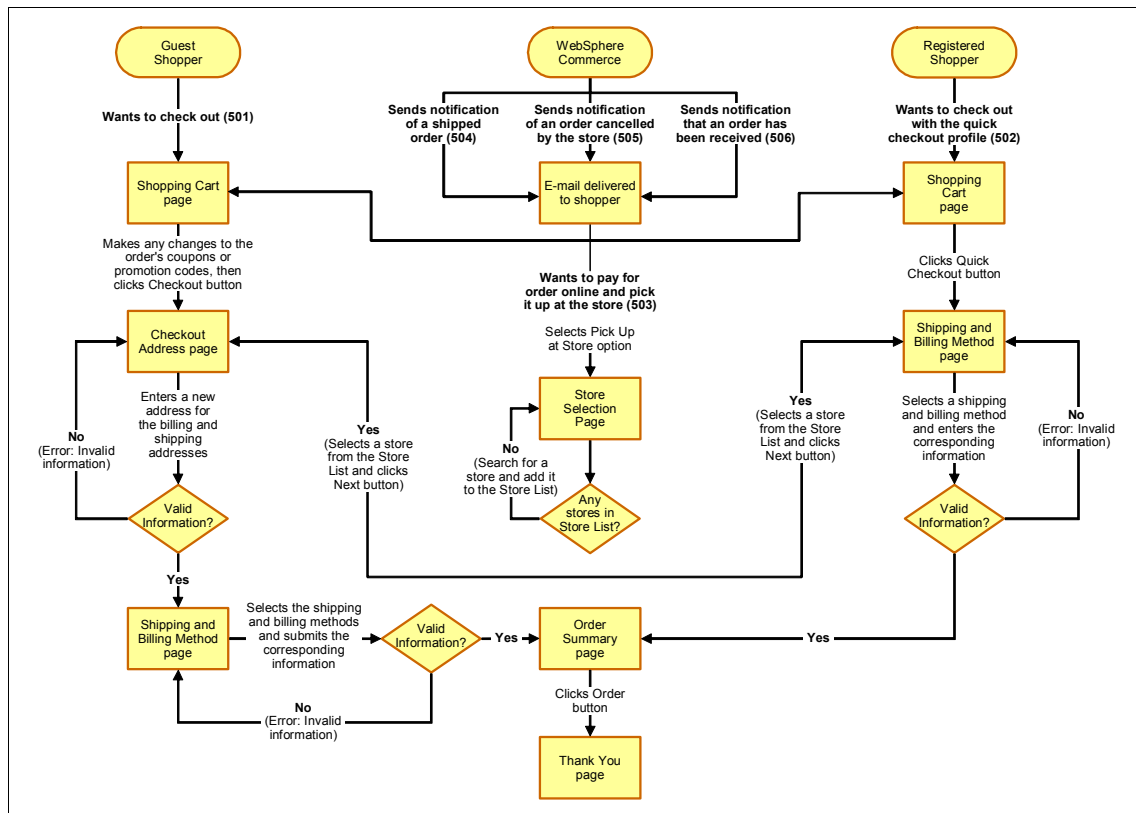


Figure 5-11 Checkout summary diagram

5.4.3 Actions and subsystem in the scenario

This section provides an overview of the actions and events triggered when the user follows the checkout flow. In our case, the channel is an online storefront. We are using the Madison store for the testing scenario. WebSphere Commerce handles this storefront operation. It interacts with the backend to reserve inventory, as discussed in 5.1, “Scenario introduction” on page 80. Table 5-4 provides a summary to help you understand what actions and systems are involved in this activity. Table 5-4 also provides a link for further details at the WebSphere Commerce Infocenter.

Table 5-4 Action and subsystem in checkout process

Overview	A guest shopper accesses the store, adds at least one item to the shopping cart, and checks out.
Channel	Online storefront.
Actor/system	Guest shopper, WebSphere Commerce.
Subsystem	Order subsystem.
Trigger event	Guest shopper wants to check out.
Additional information	http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.madisons-starterstore.doc/refs/rsmmadisoncheckoutflow.htm

5.4.4 Scenario overview and system impact results

This section discusses the checkout process flow. After the items are added to the cart, the shopper proceeds to this flow. Table 5-5 details the step-by-step checkout process.

Table 5-5 Check out process flow

Action	Response
1. The guest shopper leaves the default Shop Online option selected. 2. The guest shopper clicks Checkout.	The Shipping and Billing address page and Billing method page open.

Action	Response
3. The guest shopper enters valid information for all required fields of the billing address and different valid information for the shipping address. The guest shopper uses default shipping information for the shipping address and shipping method (Mail) and chooses the Ship as complete option. The guest shopper uses default number of payment methods (1) and default billing address, and selects the Cash on delivery billing method. 4. The guest shopper clicks Next.	The Order Summary page opens.
5. The guest shopper reviews the information on the page. 6. The guest shopper clicks Order.	The Order Confirmation page opens.

During a WebSphere Commerce order capture flow, the order component interacts with the inventory component in the following ways:

- ▶ The order component interacts with the inventory component to check the inventory availability of items in the shopping cart.
- ▶ The order component interacts with the inventory component to reserve inventory for the shopping cart when it is preparing the shopping cart for submission (that is, OrderPrepare).

The order component will most typically be configured for the inventory cache. WebSphere Commerce keeps the inventory level in cache, so the changes on the page are based on the inventory availability in the cache. The threshold is set as per the requirement of the customer who provides such a solution. In case of inventory requested going beyond threshold value, it checks the inventory level with SSFS and updates cache accordingly. If cache is not involved, a real-time call is made to SSFS. See Chapter 8, “Integration implementation” on page 195, for more information.

5.4.5 Execution flow

This section discusses the steps of the checkout flow, including exception flows, to understand the shopper's view while interacting with WebSphere Commerce integrated with SSFS. After the step Add to cart is done, the shopper reaches the checkout page and takes the following steps:

1. Click **Check out**. Figure 5-12 shows the initial page where the shopper starts the checkout process. The Shop online option is selected by default here, and you can proceed further by clicking **Checkout**.

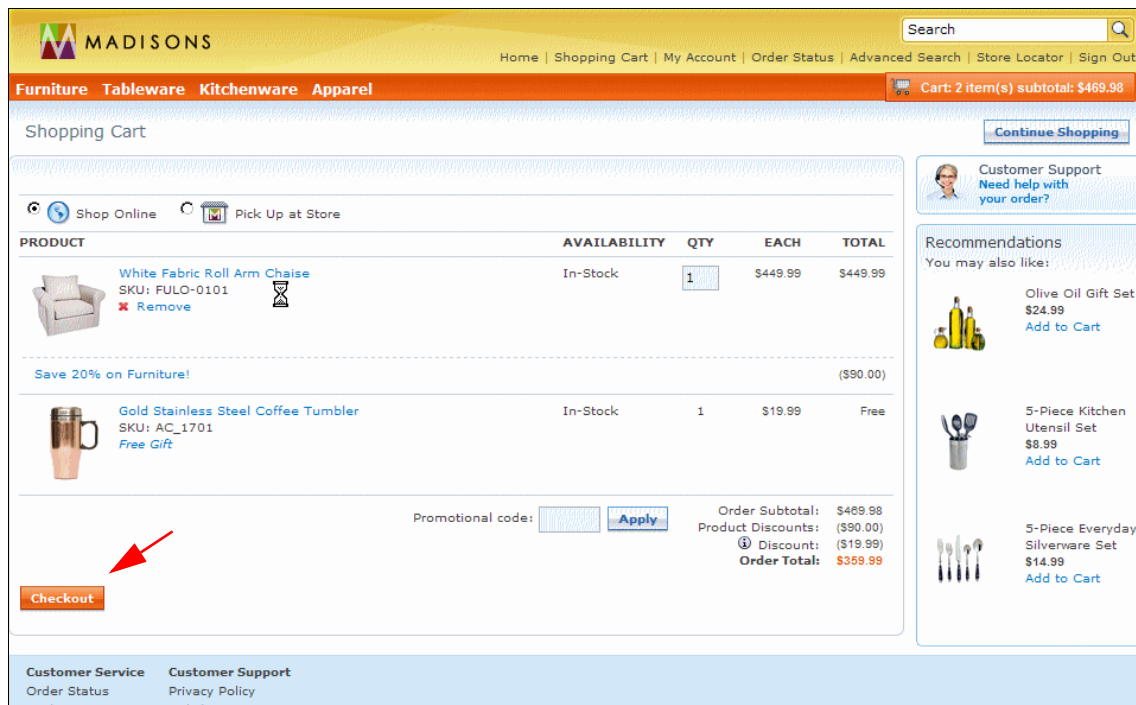


Figure 5-12 Checkout initial window

After the shopper clicks Check out, the page shown in Figure 5-13 (top portion of the page) appears requesting more details, such as the shipping and billing address and shipping and billing method information.

MADISON'S

Home | Shopping Cart | My Account | Order Status | Advanced Search | Store Locator | Sign Out

Furniture Tableware Kitchenware Apparel

Cart: 2 item(s) subtotal: \$469.98

Shopping Cart | **Shipping & Billing Method** | Order Summary

Shipping Information


Click multiple shipments if shipping to more than one address **Multiple Shipments**

Shipping Address:
wcsadmin
123 xyz circle
xyzcity North Carolina
United States 27713
xyz@in.ibm.com

Shipping Method:
US - Regular Delivery

☒ Ship as Complete
☐ Add Shipping Instructions
☐ Request Shipping Date

[Edit Address](#)
[Create Address](#)

Product	Availability	Qty	Each	Total
 White Fabric Roll Arm Chaise SKU: FULO-0101	Out of Stock	1	\$449.99	\$449.99

[Remove](#)

Figure 5-13 Billing and shipping address and method page (part1)

2. Figure 5-14 shows the remaining portion of the page. After entering the shipping and billing information, click **Next** to proceed to the Order Summary page.

Promotional code: [Apply](#)

Order Subtotal: \$469.98
Product Discounts: (\$90.00)
Discount: (\$21.92)
Tax: \$18.00
Shipping: \$8.93
Shipping Tax: \$0.00
Order Total: \$382.99

Billing Information

Select the number of payment methods: 1 Payment method(s)

BILLING ADDRESS:
wcsadmin
123 xyz circle
xyzcity North Carolina
United States 27713
xyz@in.ibm.com

BILLING METHOD:
Select billing method

*Amount: 382.99

[Edit Address](#)
[Create Address](#)

[Back](#) [Next](#) Proceed to your Order Summary.

Figure 5-14 Billing and shipping address and method page (part 2)

3. Select **Cash on delivery** method (Figure 5-15). Click **Next**. In our test scenario, we select this method. However, you can select any of the available methods according to the shopper's requirement.

The screenshot displays the 'Billing Information' section of an e-commerce checkout process. At the top right, a summary of order costs is shown: Order Subtotal (\$469.98), Product Discounts (\$90.00), Discount (\$21.92), Tax (\$18.00), Shipping (\$6.93), and Shipping Tax (\$0.00), resulting in an Order Total of \$382.99. The main form area has a yellow header 'Billing Information'. Below it, a dropdown menu shows '1 Payment method(s)'. The 'BILLING ADDRESS' section includes a dropdown for 'wcsadmin' and a text area with the address: 'wcsadmin, 123 xyz circle, xyzcity North Carolina, United States 27713, xyz@in.ibm.com'. Links for 'Edit Address' and 'Create Address' are provided. The 'BILLING METHOD' dropdown is highlighted with a red box and shows 'Cash on delivery' selected. Below this, the amount '382.99' is displayed. At the bottom, there are 'Back' and 'Next' buttons, with a red arrow pointing to the 'Next' button. The text 'Proceed to your Order Summary.' is visible next to the 'Next' button.

Promotional code: [Apply](#)

Order Subtotal: \$469.98
Product Discounts: (\$90.00)
④ Discount: (\$21.92)
Tax: \$18.00
Shipping: \$6.93
Shipping Tax: \$0.00
Order Total: \$382.99

Billing Information

Select the number of payment methods: 1 Payment method(s)

BILLING ADDRESS: wcsadmin
wcsadmin
123 xyz circle
xyzcity North Carolina
United States 27713
xyz@in.ibm.com
[Edit Address](#)
[Create Address](#)

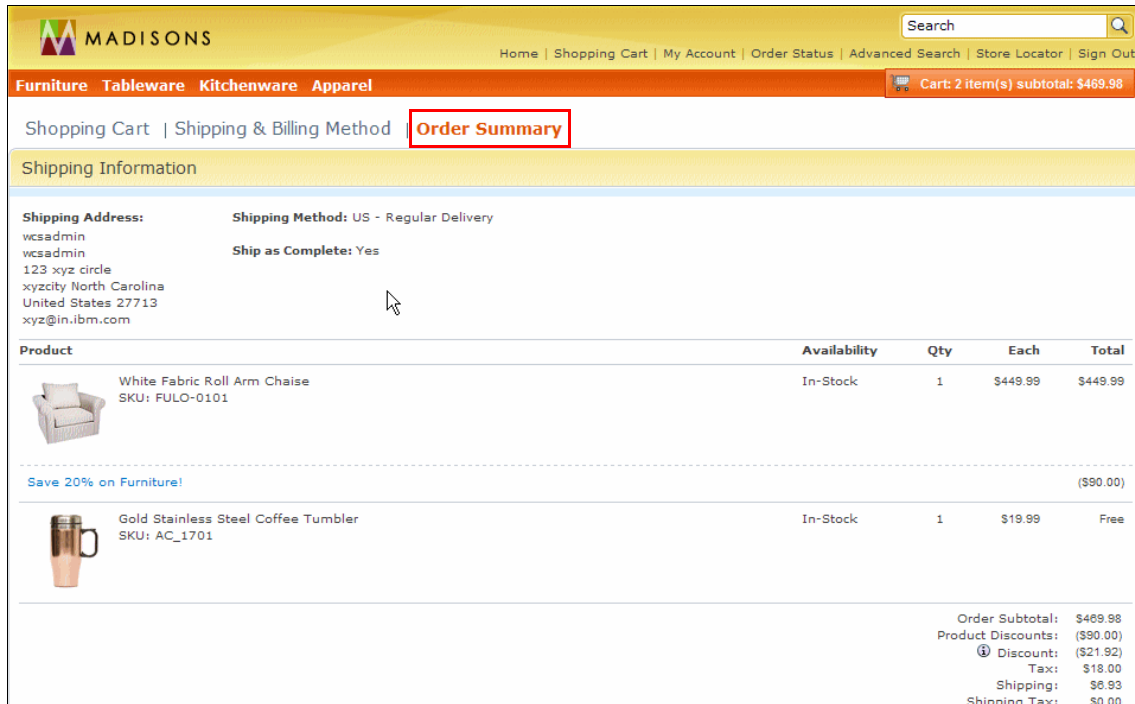
BILLING METHOD: Cash on delivery

* Amount: 382.99

[Back](#) [Next](#) Proceed to your Order Summary.

Figure 5-15 Cash on delivery method selection

- The shopper now sees the Order Summary page (Figure 5-16).The shopper reviews the information.



MADISON'S Home | Shopping Cart | My Account | Order Status | Advanced Search | Store Locator | Sign Out



Furniture Tableware Kitchenware Apparel Cart: 2 item(s) subtotal: \$469.98

Shopping Cart | Shipping & Billing Method | **Order Summary**

Shipping Information

Shipping Address:
 wcsadmin
 wcsadmin
 123 xyz circle
 xyzcity North Carolina
 United States 27713
 xyz@in.ibm.com


Shipping Method: US - Regular Delivery
Ship as Complete: Yes

Product	Availability	Qty	Each	Total
 White Fabric Roll Arm Chaise SKU: FULO-0101	In-Stock	1	\$449.99	\$449.99
Save 20% on Furniture!				(\$90.00)
 Gold Stainless Steel Coffee Tumbler SKU: AC_1701	In-Stock	1	\$19.99	Free

Order Subtotal: \$469.98
 Product Discounts: (\$90.00)
 ① Discount: (\$21.92)
 Tax: \$18.00
 Shipping: \$6.93
 Shipping Tax: \$0.00

Figure 5-16 Order summary page

Figure 5-17 shows the remaining section of the Order Summary page.

	Gold Stainless Steel Coffee Tumbler SKU: AC_1701	In-Stock	1	\$19.99	Free
---	---	----------	---	---------	------

Order Subtotal: \$469.98
Product Discounts: (\$90.00)
④ Discount: (\$21.92)
Tax: \$18.00
Shipping: \$6.93
Shipping Tax: \$0.00
Order Total: \$382.99

Billing Information

BILLING ADDRESS
wcsadmin wcsadmin
123 xyz circle
xyzcity North Carolina
United States 27713
xyz@in.ibm.com

BILLING METHOD
Cash on delivery

Amount:
\$382.99

Back

Order

Figure 5-17 Order summary page (part 2)

5. Click **Order**. The order is sent to WebSphere Commerce to process further by interacting with SSFS. Figure 5-18 shows the order confirmation page.

Thank you for your order!
You will receive a confirmation by e-mail to verify your order.
Order number: 11541
Order date: November 29, 2010

Continue Shopping

Shipping Information

Shipping Address:
wcsadmin wcsadmin
123 xyz circle
xyzcity North Carolina
United States 27713
xyz@in.ibm.com

Ship as Complete: Yes
Shipping Method: US - Regular Delivery



Product	Availability	Qty	Each	Total
 White Fabric Roll Arm Chaise SKU: FULO-0101	In-Stock	1	\$449.99	\$449.99
Save 20% on Furniture!				(\$90.00)
 Gold Stainless Steel Coffee Tumbler SKU: AC_1701	In-Stock	1	\$19.99	Free
Order Subtotal:				\$469.98
Product Discounts:				(\$90.00)
④ Discount:				(\$21.92)
Tax:				\$18.00
Shipping:				\$6.93
Shipping Tax:				\$0.00
Order Total:				\$382.99

Figure 5-18 Order confirmation page (part 1)

Figure 5-19 shows the order confirmation page.

Billing Information

BILLING ADDRESS
wcsadmin wcsadmin
123 xyz circle
xyzcity North Carolina
United States 27713
xyz@in.ibm.com

BILLING METHOD
Cash on delivery
Amount: \$382.99

Print We recommend you print this page

Order Subtotal: \$469.98
Product Discounts: (\$90.00)
④ Discount: (\$21.92)
Tax: \$18.00
Shipping: \$6.93
Shipping Tax: \$0.00
Order Total: \$382.99

Figure 5-19 Order confirmation page (part 2)

The shopper logs out and closes the browser, concluding the order capture scenario.

5.4.6 Flow verification

When the order is placed, it changes inventory within SSFS. This section covers the same scenario in which the shopper is buying three lounge chairs. After you click Check out and click Next, the order goes to the SSFS, where inventory gets reserved. Follow the steps below to check the inventory in SSFS.

1. Perform the following preparation steps to get to the point of looking at the inventory:
 - a. Log in to the SSFS application console using following URL:
`http://<HOSTNAME>/smcfs/console/login.jsp`
 - b. Click the **Inventory** tab and select **Inventory Console**.
 - c. On the left panel, select the organization and click **Search**.
 - d. Click the item for which you want to check inventory (in our case, FULO-0101). Now you are at the page shown in Figure 5-20, which shows the initial quantity at of the SSFS application console. As you can see, the reserved quantity is currently 32.

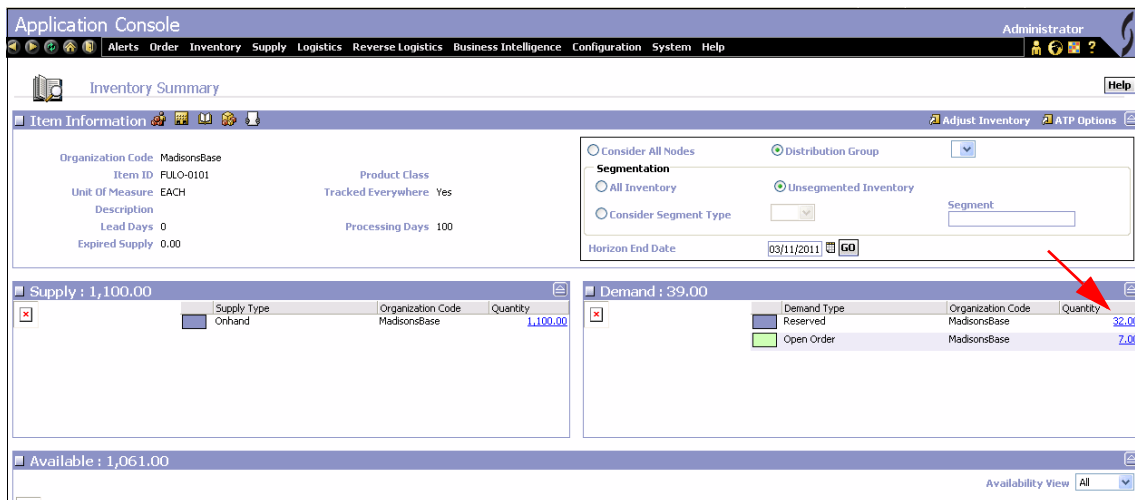


Figure 5-20 Initial SSFS application console panel

2. Click **Add to cart** and proceed to checkout. Refer to the example flow discussed in 5.4.5, “Execution flow” on page 95. This reserves inventory in SSFS. Figure 5-21 shows this transaction, where three items are added to a cart.

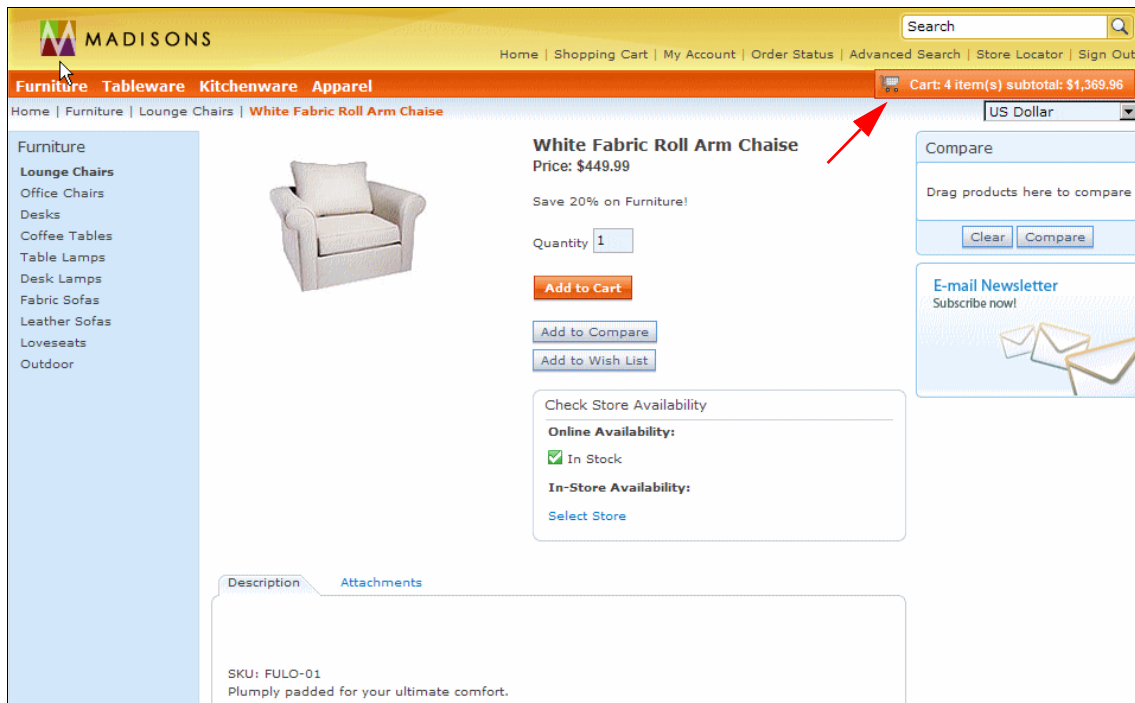



Figure 5-21 Add items to a cart

3. Click **Checkout** → **Next** → **Next** → **Order**. Figure 5-22 shows that the item is added to the cart. After you click **Checkout** and **Next**, the order is passed to SSFS to reserve the quantity.



Shopping Cart

Continue Shopping

Customer Support
Need help with your order?

Shop Online

Pick Up at Store

PRODUCT	AVAILABILITY	QTY	EACH	TOTAL
<div><div><div>White Fabric Roll Arm Chaise</div><div>SKU: FULO-0101</div><div><div>Remove</div></div></div></div> <div>In-Stock</div> <div><div>1</div></div> <div>\$449.99</div> <div>\$1,349.97</div>				
Save 20% on Furniture!				(\$269.99)
<div><div><div>Gold Stainless Steel Coffee Tumbler</div><div>SKU: AC_1701</div><div>Free Gift</div></div></div> <div>In-Stock</div> <div>1</div> <div>\$19.99</div> <div>Free</div>				

Promotional code:

Apply

Order Subtotal: \$1,369.98


Product Discounts: (\$269.99)

Discount: (\$23.90)


Order Total: \$1,076.07

Checkout


You may also like:



Olive Oil Gift Set
\$24.99
[Add to Cart](#)



5-Piece Kitchen Utensil Set
\$8.99
[Add to Cart](#)



5-Piece Everyday Silverware Set
\$14.99
[Add to Cart](#)

Figure 5-22 Checkout page

Chapter 5. Business scenario: Order capture 103

- Go back to the SSFS application console. To refresh the page, press F5. The SSFS application console inventory level changes. Figure 5-23 shows the quantity after the order is placed.

The screenshot shows the 'Application Console' interface. The top navigation bar includes 'Alerts', 'Order', 'Inventory', 'Supply', 'Logistics', 'Reverse Logistics', 'Business Intelligence', 'Configuration', 'System', and 'Help'. The 'Inventory Summary' section is active, showing 'Item Information' for 'MadsonsBase' with 'Item ID' FULO-0101. The 'Supply' section shows a total of 1,100.00. The 'Demand' section shows a total of 42.00, with a table listing 'Reserved' (35.00) and 'Open Order' (7.00) demand types. A red arrow points to the 'Reserved' row in the demand table.

Supply Type	Organization Code	Quantity
Onhand	MadsonsBase	1,100.00

Demand Type	Organization Code	Quantity
Reserved	MadsonsBase	35.00
Open Order	MadsonsBase	7.00

Figure 5-23 Inventory change after order is placed

For example, you clicked Checkout and Next. When you go back and look at the SSFS application console, you can see that three more items are added in the reserved inventory. Now the quantity shows 35 items. In this case, WebSphere Commerce makes a call to WebSphere Enterprise Service Bus, which in turn transforms the message and passes to SSFS to reserve inventory. The mediation module in WESB transforms the request from WebSphere Commerce and passes it to SSFS to serve the request. Further order processing activities are handled by SSFS.

Note: There could be a delay in passing the order from WebSphere Commerce due to latency. The order might not reflect the change immediately in the system. To change the latency:

1. Open the IBM WebSphere Commerce Administration Console and log on:
<https://<HOSTNAME>:8002/webapp/wcs/admin/servlet/ToolsLogon?XMLFile=adminconsole.AdminConsoleLogon>
2. Select **Site** and click **OK**.
3. Select **Schedulers** from the Configurations menu.
4. In the table, select **SendTransactedMsg** commands and click **Change**, which opens a new window with configurable parameters.
5. Modify the value of the text box labeled Schedule interval.
6. Save the changes.

5.5 Buy-online-pickup-in-store (BOPIS) scenario

This section discusses two scenarios:

- Buy-online-and-pick-up-in-store (pay in store unchecked)
- Buy-online-and-pay-in-store (pay in store checked)

During the order capture process, the shopper can use these scenario if they want to pick up an item from the store. This functionality is provided OOB by WebSphere Commerce.

Figure 5-24 shows that the online store is out of stock. In this case, you can select a store from which to pick up item, and either pay online or pay in store.

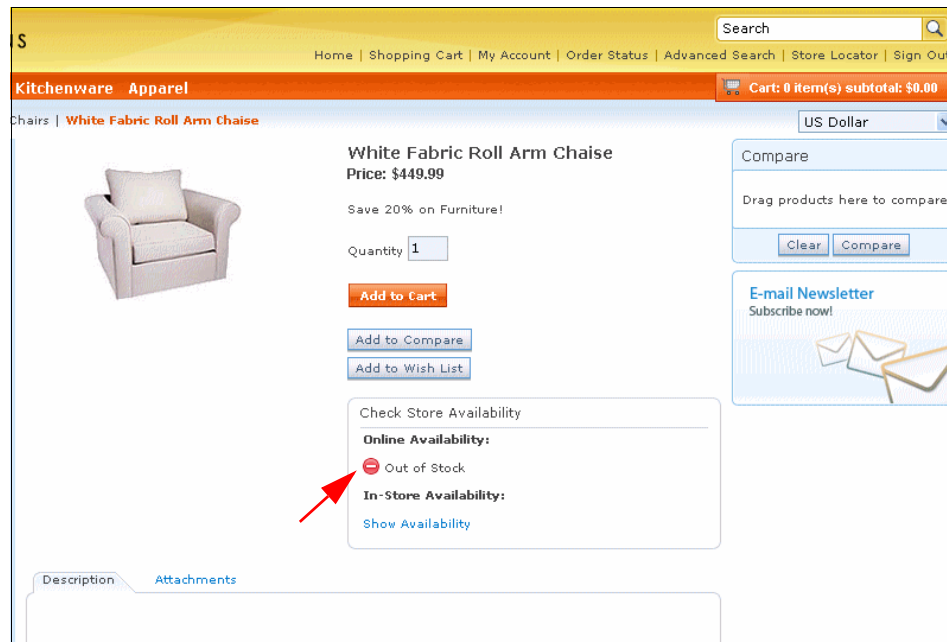


Figure 5-24 Item out of stock in online store

For the buy-online-and-pick-up-in-store scenario, perform the following steps:

1. Click **Add to cart**, adding the item to the cart. You can proceed to checkout now. See 5.4.5, “Execution flow” on page 95, for details. Figure 5-25 shows the checkout page. Notice that you have two options:
 - Online store
 - Pick up in store

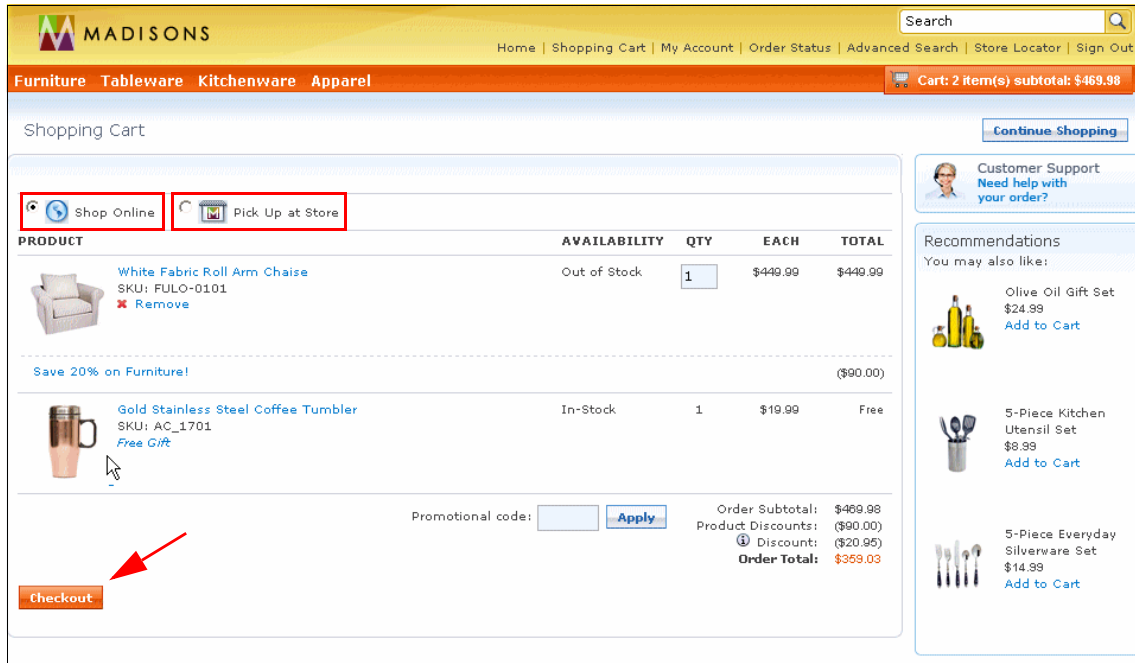


Figure 5-25 Add item to cart and proceed to checkout

2. Select **Pick up in store** (Figure 5-26). If you had selected online store instead, the page would show an error message stating that there is insufficient inventory.

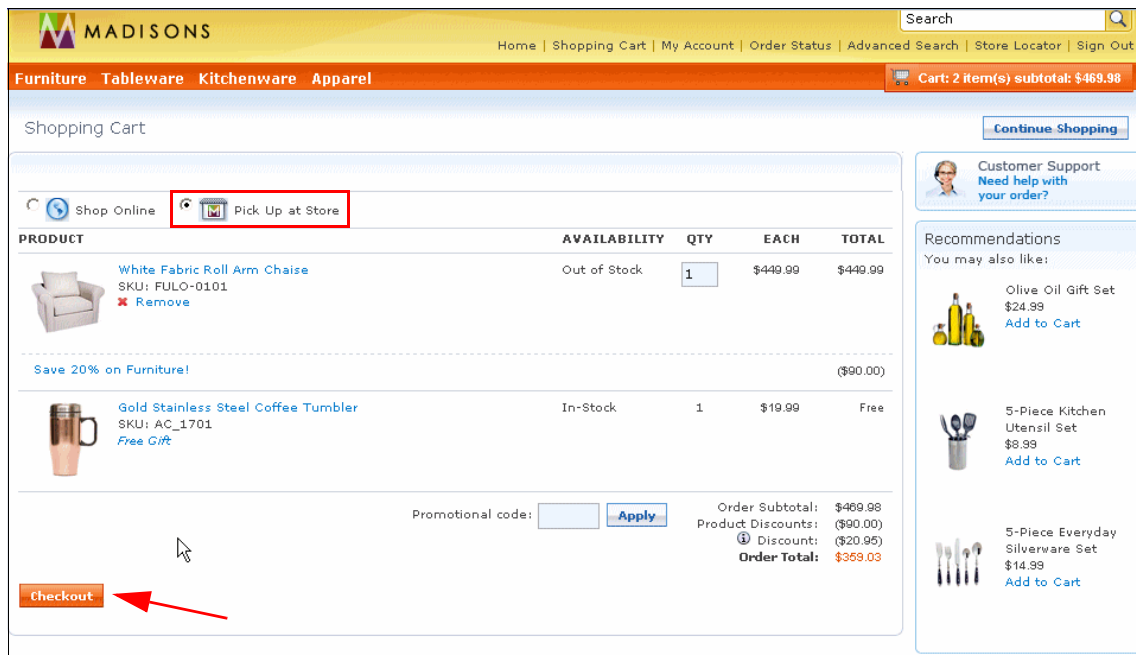
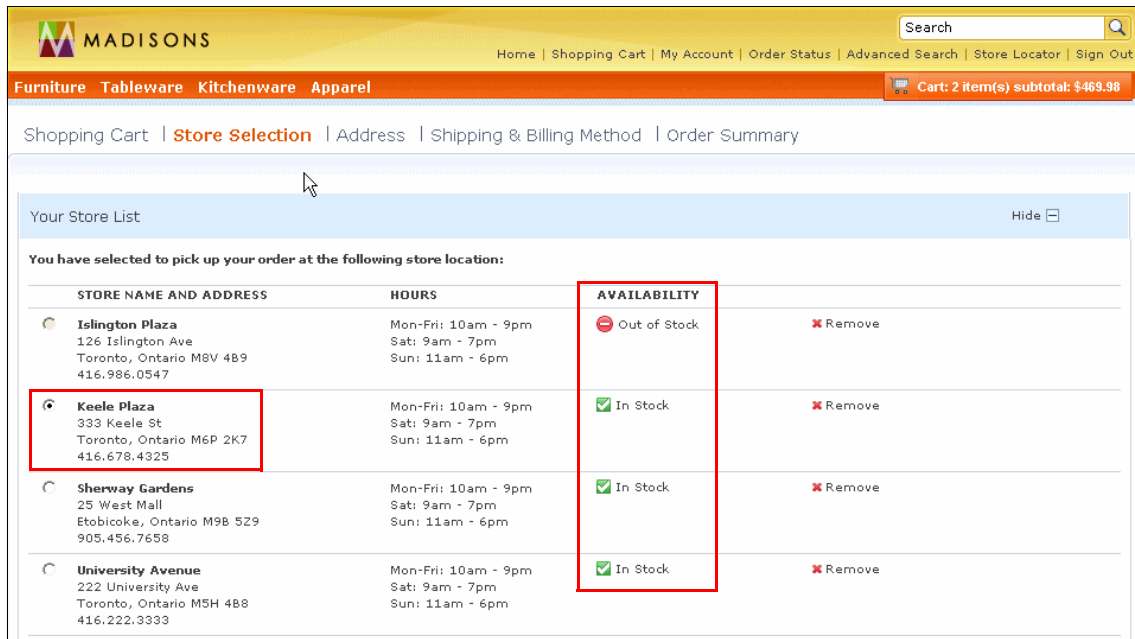


Figure 5-26 Select pick up in store

3. Click **Check out**. Figure 5-27 shows a list of stores available where you can pick up the item. As you can see, a few stores show the item as out of stock, and others show the item as in stock. You can select a store with the in stock status and proceed further for order placement. In this case, select **Keele Plaza**.



The screenshot shows the MADISON'S website interface. At the top, there is a navigation bar with links for Home, Shopping Cart, My Account, Order Status, Advanced Search, Store Locator, and Sign Out. Below this is a secondary navigation bar with categories: Furniture, Tableware, Kitchenware, and Apparel. A cart summary indicates 'Cart: 2 item(s) subtotal: \$469.98'. The main content area is titled 'Store Selection' and shows a list of stores available for pickup. The list is organized into columns: STORE NAME AND ADDRESS, HOURS, AVAILABILITY, and a Remove button. The 'Keele Plaza' store is highlighted with a red box, indicating it is the preferred store for pickup. The 'Islington Plaza' store is marked as 'Out of Stock', while the other three stores are marked as 'In Stock'.

STORE NAME AND ADDRESS	HOURS	AVAILABILITY	
Islington Plaza 126 Islington Ave Toronto, Ontario M8V 4B9 416.986.0547	Mon-Fri: 10am - 9pm Sat: 9am - 7pm Sun: 11am - 6pm	Out of Stock	Remove
Keele Plaza 333 Keele St Toronto, Ontario M6P 2K7 416.678.4325	Mon-Fri: 10am - 9pm Sat: 9am - 7pm Sun: 11am - 6pm	In Stock	Remove
Sherway Gardens 25 West Mall Etobicoke, Ontario M9B 5Z9 905.456.7658	Mon-Fri: 10am - 9pm Sat: 9am - 7pm Sun: 11am - 6pm	In Stock	Remove
University Avenue 222 University Ave Toronto, Ontario M5H 4B8 416.222.3333	Mon-Fri: 10am - 9pm Sat: 9am - 7pm Sun: 11am - 6pm	In Stock	Remove

Figure 5-27 Select preferred store

Click **Next**. Figure 5-28 shows the option to pay online or pay in store. Leave Pay in store unchecked.

MADISON'S

Home | Shopping Cart | My Account | Order Status | Advanced Search | Store Locator | Sign Out

Furniture Tableware Kitchenware Apparel

Cart: 2 item(s) subtotal: \$469.98

Shopping Cart | Store Selection | **Address** | Shipping & Billing Method | Order Summary

1. Billing Address

☐ Pay in-store

The billing address is selected on the next page.

2. Store Address

Keele Plaza
333 Keele St
Toronto Ontario
Canada M6P 2K7
416.678.4325

[Back](#) [Next](#) Proceed to your Shipping & Billing Method

Customer Service
Order Status
Wish List
My Account

Customer Support
Privacy Policy
Help/Contact Us
Site Map

Figure 5-28 Do not select Pay in store

Click **Next**. Because you unchecked the pay in store check box, you are provided with billing and payment method options. Figure 5-29 shows the billing and payment options.

Promotional code: [Apply](#)

Order Subtotal: \$469.98
Product Discounts: (\$90.00)
Discount: (\$19.99)
Tax: \$0.00
Shipping: \$0.00
Shipping Tax: \$0.00
Order Total: \$359.99

Billing Information

Select the number of payment methods: 1 Payment method(s)

BILLING ADDRESS: wcsadmin

123 xyz circle
xyzcity North Carolina
United States 27713
xyz@in.ibm.com

BILLING METHOD: Select billing method

* Amount: 359.99

[Edit Address](#)
[Create Address](#)

[Back](#) [Next](#) Proceed to your Order Summary.

Figure 5-29 Billing and payment options page

You can take an alternate flow and select the Pay in store check box and click **Next**. Figure 5-30 shows Pay in store selected, in which case the user proceeds to the next window.

MADISONS

Home | Shopping Cart | My Account | Order Status | Advanced Search | Store Locator | Sign Out

Furniture Tableware Kitchenware Apparel

Cart: 2 item(s) subtotal: \$469.98

Shopping Cart | Store Selection | **Address** | Shipping & Billing Method | Order Summary

1. Billing Address

☒ Pay in-store

The billing address is selected on the next page.

2. Store Address

Keele Plaza
333 Keele St
Toronto Ontario
Canada M6P 2K7
416.678.4325

[Back](#) [Next](#) Proceed to your Shipping & Billing Method

Customer Service
Order Status
Wish List
My Account

Customer Support
Privacy Policy
Help/Contact Us
Site Map

Figure 5-30 Select pay in store

- The shopper clicks **Next**. Figure 5-31 shows that there are no billing and payment options, because the shopper selected the Pay in store option.

Billing Information

Currently selected payment method:

BILLING ADDRESS:

wcsadmin

wcsadmin
123 xyz circle
xyzcity North Carolina
United States 27713
xyz@in.ibm.com

[Edit Address](#)
[Create Address](#)

[Back](#) [Next](#) Proceed to your Order Summary.

Your payment is made at the store location.

[Change](#)

Figure 5-31 Panel without billing and payment options

- Click **Next**. You see the Order Summary window. Click **Order**. This shows the order details and shows you the order successfully placed message. This concludes these scenarios.

In this chapter, we discussed how the shopping cart, BOPIS, and checkout flows work in WebSphere Commerce and during this operations how they interact with SSFS via Enterprise Service Bus infrastructure provided by WESB.



Business Scenario: Order status

Once the order is submitted successfully, the user sees the order confirmation page with the order number. Using this order number, the user is able to check the order status. The order status page lists all the orders in a particular shopper account with the order numbers for each of their orders. This status shows the different process levels that the order is currently going through.

This chapter contains the following sections:

- ▶ 6.1, “Scenario introduction” on page 114
- ▶ 6.2, “Prerequisites” on page 114
- ▶ 6.3, “Order status flow” on page 121
- ▶ 6.4, “Sterling order fulfillment” on page 127

6.1 Scenario introduction

After an order has been placed in WebSphere Commerce, the Sterling Selling and Fulfillment Suite (SSFS) Distributed Order Management (DOM) and the external system deal with the delivery of that order. All aspects of fulfilling the order, whether a matter of packing items from inventory or manufacturing items and all aspects of billing, shipping, and so, are handled by the external systems. Because the external systems will be managing the order status at a more granular level, the WebSphere Commerce picture of that order will need to be updated with the order status as it moves through the external systems in order to give an accurate representation of the status to the shopper.

WebSphere Commerce maintains a cache of the latest order status, which is updated by WebSphere Commerce during the many steps of order capture processing. After the order is submitted to the backend DOM system for processing, the order status can be updated by using data received from the backend DOM system. The choice of using the *push* or *pull* approach of updating the order status is one of the enterprise architecture decisions.

As the order flows through the various stages, the order status is updated by the external systems and the external system sends a sync order message to WebSphere Commerce to update the status of the order. WebSphere Commerce then has the latest status of the order to be used to display the current status of the order to the shopper. Additionally, WebSphere Commerce has the ability to maintain order status history for the order and the order items, providing the flexibility of communicating to the shopper the various order stages and the points in time that they occurred. However, because SSFS also has data supporting order status history (audit), and the DOM would have a more complete view of the order fulfillment process, the solution architect should use SSFS to keep the appropriate order status history record and therefore avoid maintaining redundant data.

This chapter describes a typical *push* model of order status, where the status is sent by the system as the status changes. A less common approach is to have a real-time query of the order status as needed. The possible benefit to this is that fewer messages are sent, because the only status messages required are the ones that are demanded.

6.2 Prerequisites

This section discusses the prerequisite steps and configuration required for executing an order status scenario.

Ensure that you have configured WebSphere Commerce, WebSphere Enterprise Service Bus (WESB), and SSFS as per Chapter 7, “Installation and configuration” on page 129. In addition, there are two more prerequisites at the WebSphere Commerce frontend to check the order status in the store:

- ▶ Enable the order status in the management center.
- ▶ The shopper must be a registered user.

6.2.1 Enabling the order status in the management center

Table 6-1 explains the scenario summary and systems involved to complete this scenario.

Table 6-1 Scenario summary information for order status

Overview	Administrator accesses the management center, logs in to the management center, enables the order status tracking
Channel	Online management center
Actor/system	Administrator/business user
Subsystem	WebSphere Commerce
Trigger event	Store administrator wants to enable the order status tracking
Additional information	<code>https://<host name>:8000/lobtools</code>

The steps are:

1. Launch the management center using following URL:

`https://<host name>:8000/1obtools`

This URL opens the management center login page (Figure 6-1).

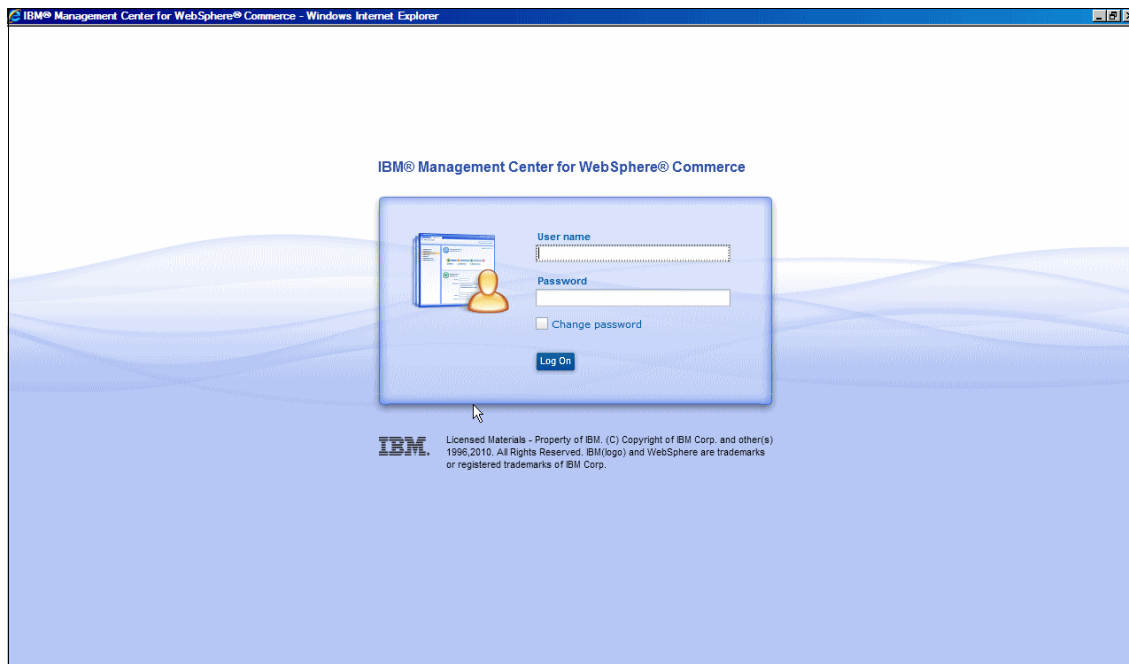


Figure 6-1 Management center login panel

2. Log in to the management center as the admin user.

The management center supports administration privilege, allowing these users to do administration on different modules of WebSphere Commerce.

A successfully login enables the user to see the management center home page.

3. In the top left corner menu on the home page (Figure 6-2), select **store management**.

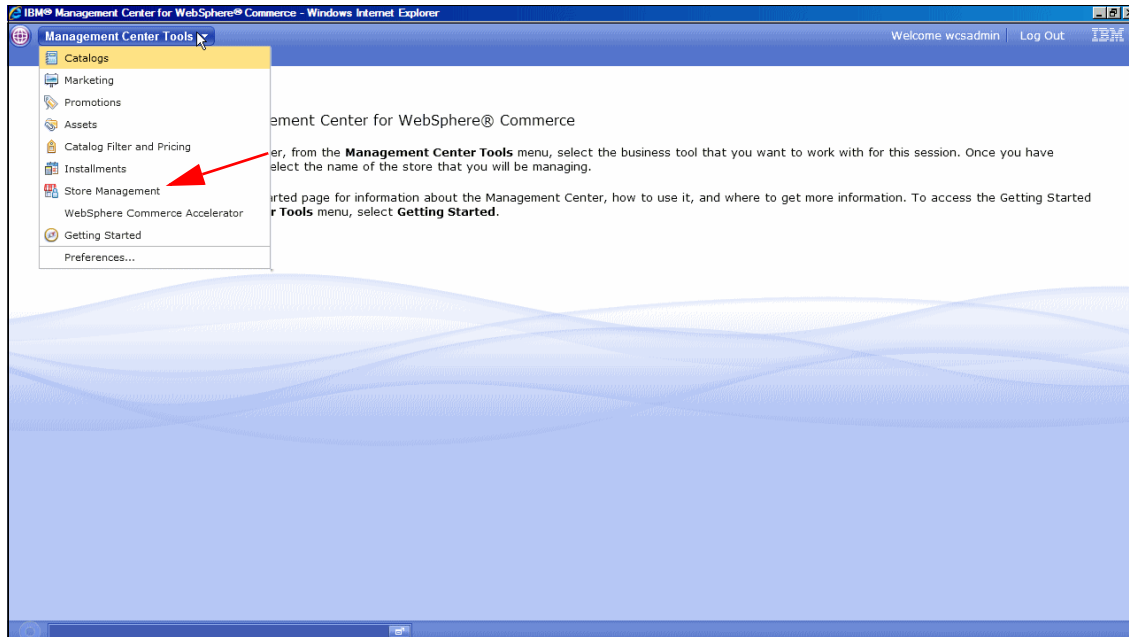


Figure 6-2 Store management module selection

4. After selecting store management from the menu, you see a list of stores on the left side bar (Figure 6-3). Click **Madisons** from the list.

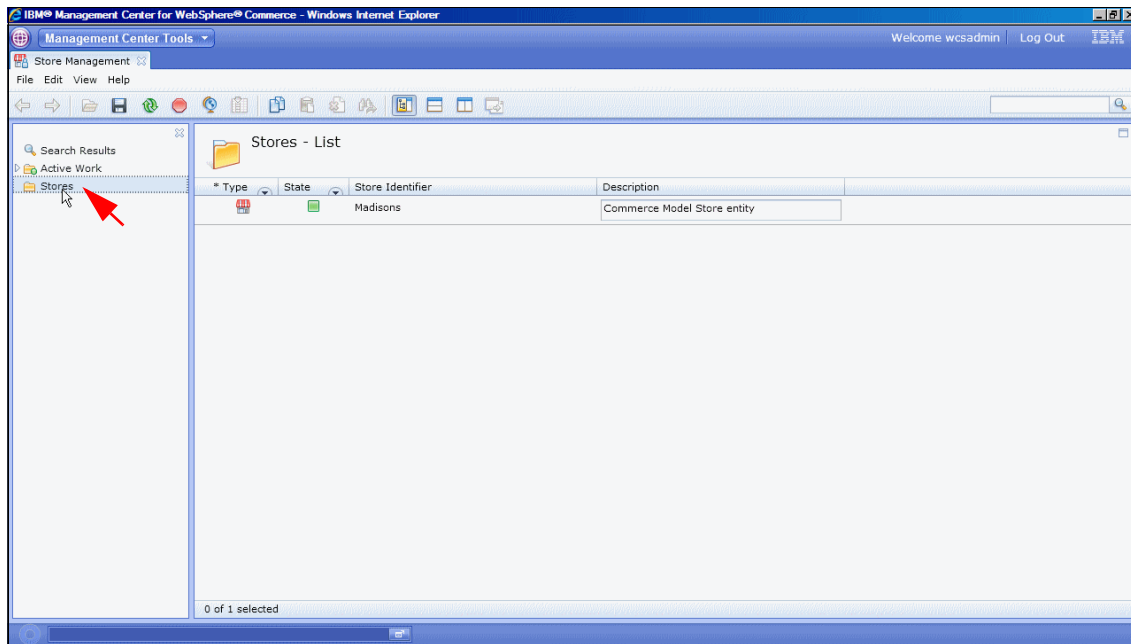


Figure 6-3 Store selection

5. Right-click the store and select **Open** (Figure 6-4), which opens the store for further updates.

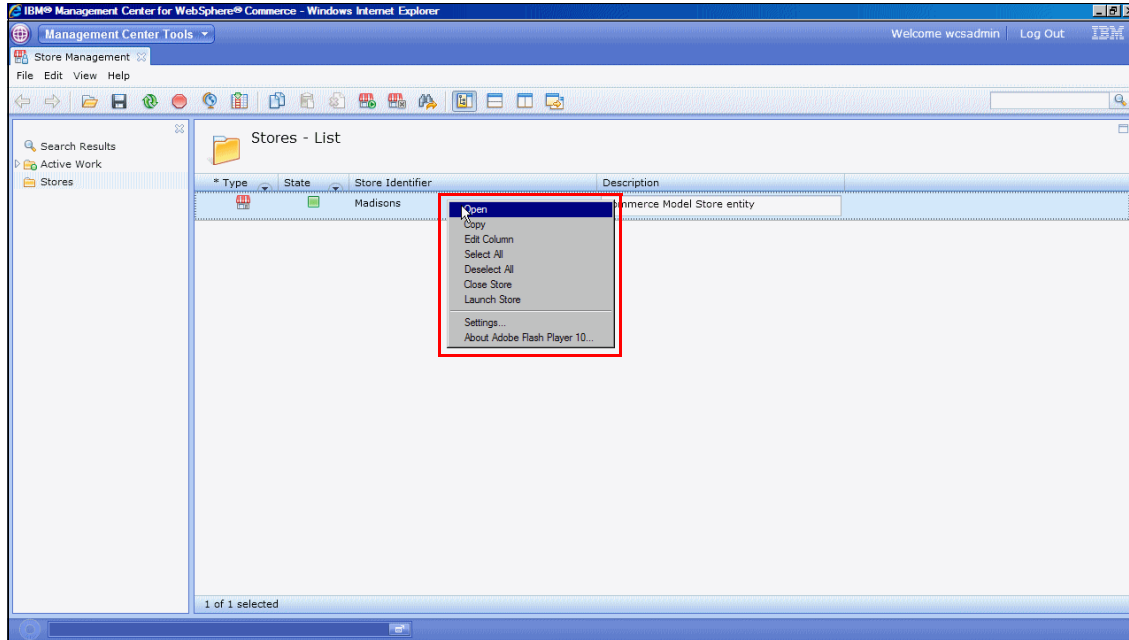


Figure 6-4 Open store for updates

6. Click the **Orders** tab (Figure 6-5).

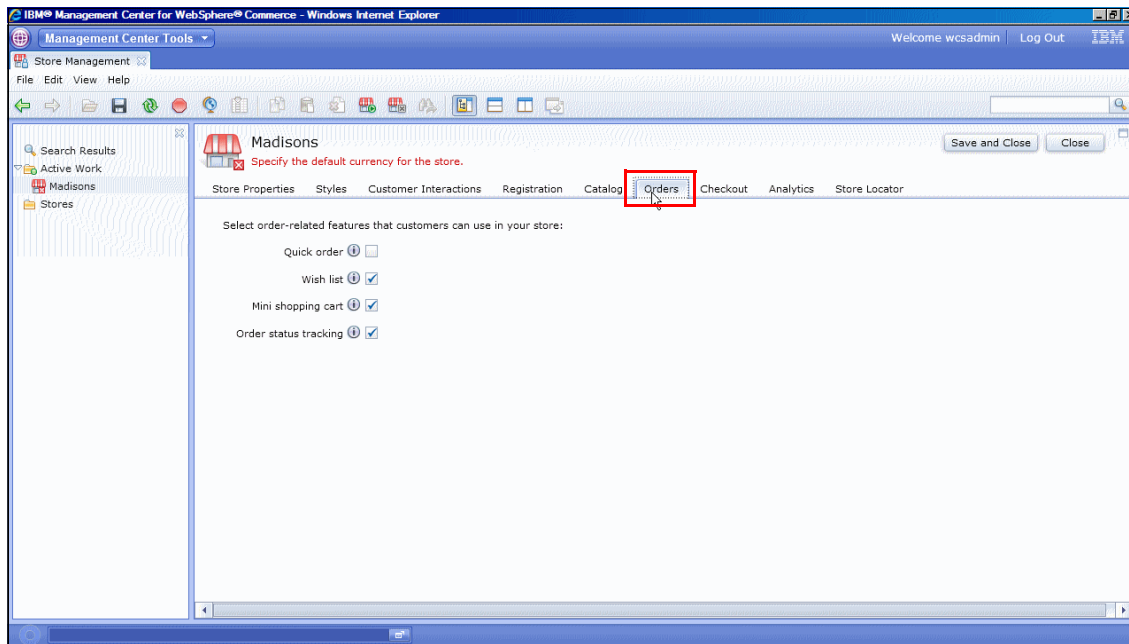


Figure 6-5 Orders tab selection

7. Check the order status on the list and save the changes (Figure 6-6), enabling the order status on the storefront.

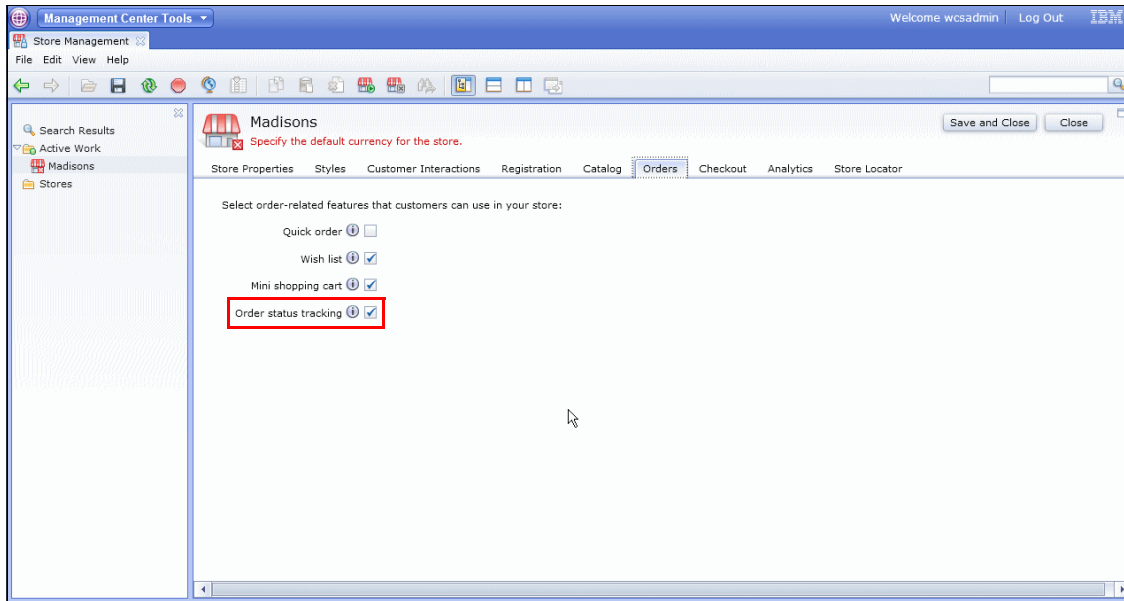


Figure 6-6 Select Order status tracking

6.2.2 User should be a registered user

If the user is a guest user, he will be able to place a guest order but will not be able to see the order history. To track the order history, the user needs to log in to the store before placing an order or needs to register at the end of the shopping flow.

6.3 Order status flow

After taking the above steps, a registered user will be able to see the order status in the storefront. The following sections describe how the order status flow works on storefront.

6.3.1 System interaction diagram

Figure 6-7 shows an architectural flow for the order status between different systems. It shows frontend interactions with the WebSphere Commerce system and the synchronization process between WebSphere Commerce and SSFS systems.

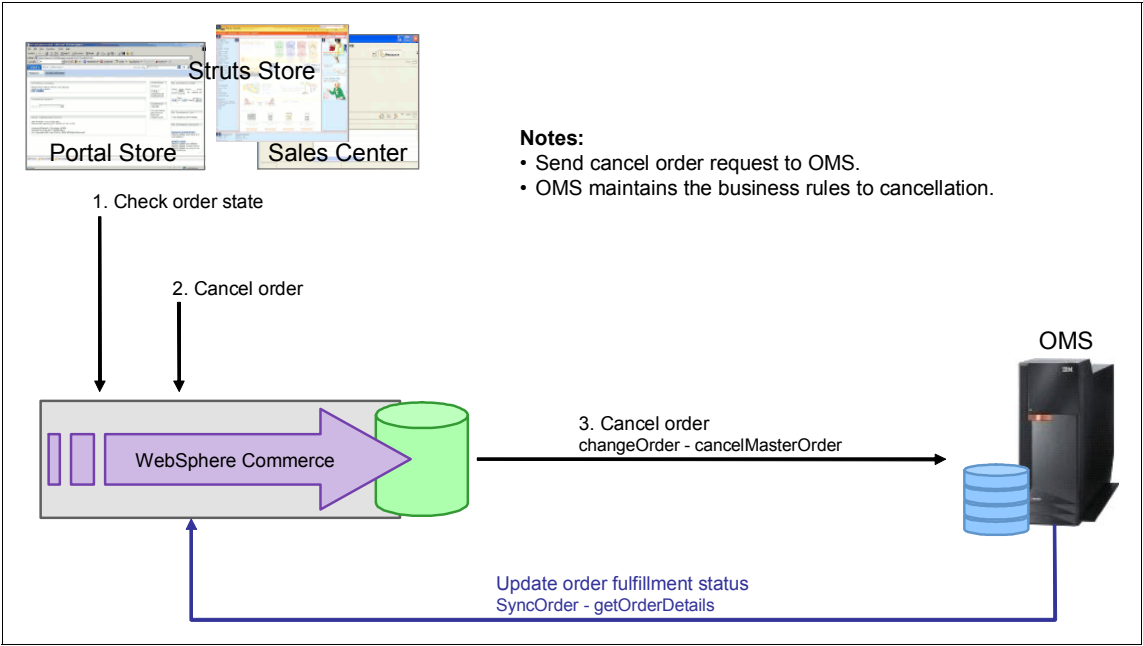


Figure 6-7 Order status flow diagram

6.3.2 Actions and subsystems in the scenario

Table 6-2 summarizes the scenario and the systems are involved to complete this scenario.

Table 6-2 Scenario summary information for order status

Overview	A shopper accesses the store, logs in to the storefront, and checks the previous orders and status.
Channel	Online storefront.
Actor/system	Shopper.
Subsystem	WebSphere Commerce, SSFS.
Trigger event	Shopper wants to check order status.

Overview	A shopper accesses the store, logs in to the storefront, and checks the previous orders and status.
Additional information	Make sure that the order status tracking is enabled from the management center.

6.3.3 Execution flow

To check the order status the user should be a registered user. The steps below take you through complete flow.

1. Launch the store using the following store URL:
`http://<hostname>/webapp/wcs/stores/servlet/TopCategories_<storeid>_<catalogid>`
This takes you to the store home page (Figure 6-8).
2. Click **Sign in** from the top menu bar (upper right corner).

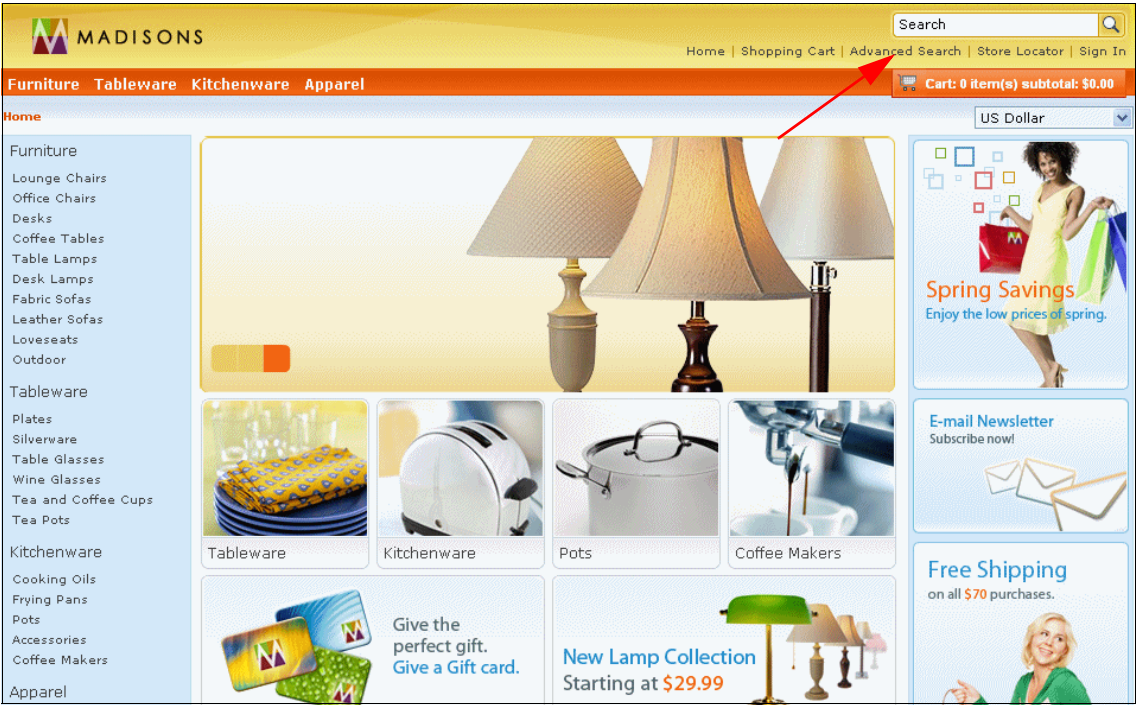


Figure 6-8 Store home page

3. If you have previously registered with the store, use the credentials and log in. Otherwise, click **Register** to create a new account (Figure 6-9).

MADISONS Search

Home | Shopping Cart | Advanced Search | Store Locator | Sign In

Furniture Tableware Kitchenware Apparel **Cart: 0 item(s) subtotal: \$0.00**

Sign In or create a new Madisons account profile.

Returning Customer

Logon ID:

Password:

☐ Remember Me
[Forgot your password?](#)

Sign In

New Customer

Registering provides you with personalized services, including:

- > Quick checkout
- > Wish list
- > Advance notice on our promotions
- > Order status
- > Personal address book

Register

Customer Service
Order Status
Wish List
My Account

Customer Support
Privacy Policy
Help/Contact Us
Site Map

Figure 6-9 Store login page

After successful login, you are taken to the MyAccount page Figure 6-10.

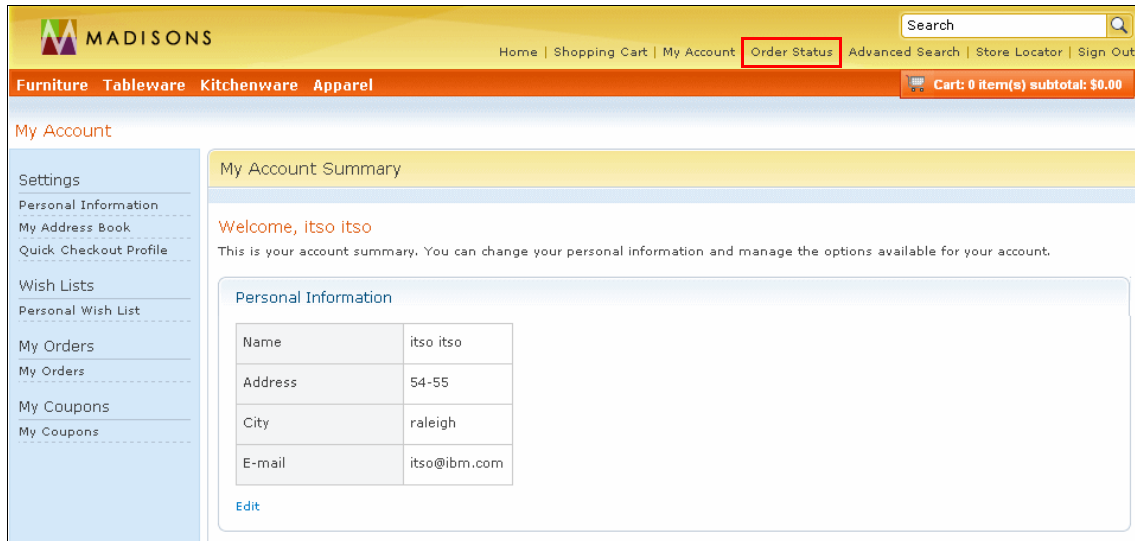


Figure 6-10 Store My Account Summary page

4. Click **Order status** on the top menu bar. This takes you to the order status page (Figure 6-11). This page shows the order status information, which is synchronized from the SSFS system.

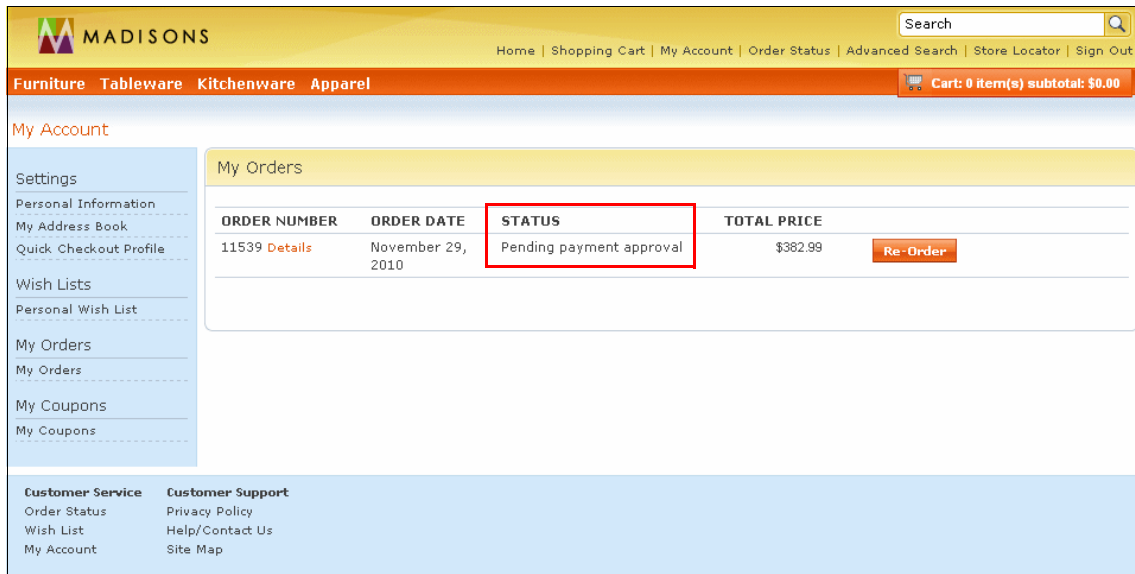


Figure 6-11 Store order status page

Note: An offline synchronization runs periodically to update the WebSphere Commerce system with the latest order information from SSFS. WebSphere Commerce shows the latest synchronized order status to the users.

After the order is synchronized from SSFS, the status changes (Figure 6-12).

The screenshot shows the MADISON'S website interface. At the top, there is a navigation bar with links: Home | Shopping Cart | My Account | Order Status | Advanced Search | Store Locator | Sign Out. Below this is a search bar and a cart summary: Cart: 0 item(s) subtotal: \$0.00. The main content area is titled 'My Account' and includes a sidebar with links: Settings, Personal Information, My Address Book, Quick Checkout Profile, Wish Lists, Personal Wish List, My Orders, My Orders, My Coupons, and My Coupons. The main content area displays 'My Orders' with a table showing one order:

ORDER NUMBER	ORDER DATE	STATUS	TOTAL PRICE
11539 Details	November 29, 2010	Pending payment approval	\$382.99

Below the table, there is a 'Re-Order' button. At the bottom of the page, there is a 'Customer Service' section with links: Order Status, Wish List, My Account, Privacy Policy, Help/Contact Us, and Site Map.

Figure 6-12 Order status template after synchronization from Sterling Commerce system

6.4 Sterling order fulfillment

There are several scenarios in which a WebSphere Commerce order, already successfully sent to Sterling Order Management, might need to be updated with a change in order status during downstream order processing. WebSphere Commerce DOM Integration components can accommodate, at a minimum, the following for downstream order processing:

- Shipments

A notification of a shipped order, in its entirety, is normally sent back to Commerce, and the order is updated accordingly.

- Partial or split orders

After the order is submitted to Sterling, Sterling Order Management's intelligent sourcing engine looks across all locations, including external partners, to determine the best location to fulfill each line on the order, based on a wide set of parameters for each order, and seamlessly splits or consolidates order lines and sequence activities. It brokers documents and requests to the appropriate internal or external fulfillment participants, and incorporates user-defined events to effectively track fulfillment activity based on the unique conditions of each order line. This intelligent brokering of the order can result in an order that might be partially shipped and or back ordered. The requirements might dictate to reflect the order's partial, split, or back order status in WebSphere Commerce.

- Canceled orders

CSR can issue an order cancel for multiple reasons, and this order status can be routed back to WebSphere Commerce.

- Returns

Sterling Order Management has predefined business process flows, which ensures that returned products are consistently handled in the proper manner and that no items are lost or forgotten in the process. This can ensure that your organization is efficiently utilizing all inventory, thus reducing your overall inventory costs.

- In transit

Sterling Order Management provides a single comprehensive view of all inventory information by aggregating inventory from all locations and providing a view of what is available internally and at all partner locations, what is being supplied, what is in transit, and the current demand. A WebSphere Commerce view of an order can be updated with this inventory status via the order status interface.

The out-of-the-box WebSphere Commerce DOM Integration components provide support for the above order status scenarios with only changes to configuration, assuming that there are no modifications to the object or database models. As requirements greatly impact which of these integration events are captured in WebSphere Commerce, it is left to the reader to decide which are relevant to him.



Installation and configuration

This chapter discusses the details of installing and configuring with IBM Sterling Selling and Fulfillment Suite (SSFS), Distributed Order Management (DOM), WebSphere Message Broker, WebSphere Integration Developer, and WebSphere Commerce as it pertains to the Sterling Selling and Fulfillment Suite and WebSphere Commerce integration in WebSphere Commerce V7 FEP2.

In this chapter, the following topics are presented:

- ▶ 7.1, “Introduction” on page 130
- ▶ 7.2, “WebSphere Commerce installation and configuration” on page 132
- ▶ 7.3, “WESB mediation module installation and configuration” on page 133
- ▶ 7.4, “Installing, configuring, and deploying SSFS” on page 140
- ▶ 7.5, “Configuring SSFS for integration with WebSphere Commerce” on page 157
- ▶ 7.6, “Integration flow data mapping” on page 182

7.1 Introduction

In this section we discuss various aspects of integrating WebSphere Commerce and IBM Sterling Selling and Fulfillment Suite.

7.1.1 Integration overview

WebSphere Commerce has the ability to interact with an external DOM solution. With the acquisition of Sterling Commerce, IBM now has a best-of-breed DOM system and integration with the IBM SSFS is supported. You can use WebSphere Commerce DOM integration with IBM SSFS as a starting point and as building blocks for your integration scenarios.

Integration of DOM with SSFS is discussed in detail in the online WebSphere Commerce documentation in the section titled “Distributed Order Management (DOM) with Sterling Selling and Fulfillment Suite (SSFS),” located at:

<http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.sterling-integration.doc/concepts/cyasterling.htm>

Integration between WebSphere Commerce and the DOM system is handled in a WebSphere Enterprise Service Bus (WESB) mediation module. This mediation module translates the Open Applications Group Integration Specification (OAGIS) Business Object Document (BOD) messages that WebSphere Commerce uses into API calls expected by SSFS.

The main flow of data between WebSphere Commerce and SSFS is the same as with any other DOM system. Data flows through the two systems (Figure 7-1).

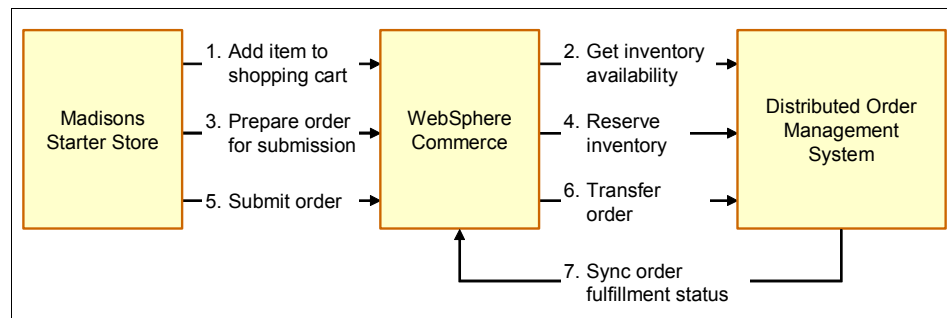


Figure 7-1 Data flow for WebSphere Commerce and SSFS integration

The mediation module that sits in between WebSphere Commerce and SSFS provides the necessary mapping for steps 2, 4, 6, and 7 in Figure 7-1. The specific mappings are discussed later in this document.

When integrating any two systems, they will have to have similar data for the integration to work. For example, when a shopper places an order in WebSphere Commerce, the products ordered have to exist in both WebSphere Commerce and SSFS (Figure 7-2). The discussion of data synchronization of this type is outside the scope of this documentation, but is a consideration.



Figure 7-2 Communications model for WebSphere Commerce and SSFS integration

For transports, the mediation module uses SOAP over HTTP(S) to communicate with WebSphere Commerce and JMS to communicate with SSFS (Figure 7-3).

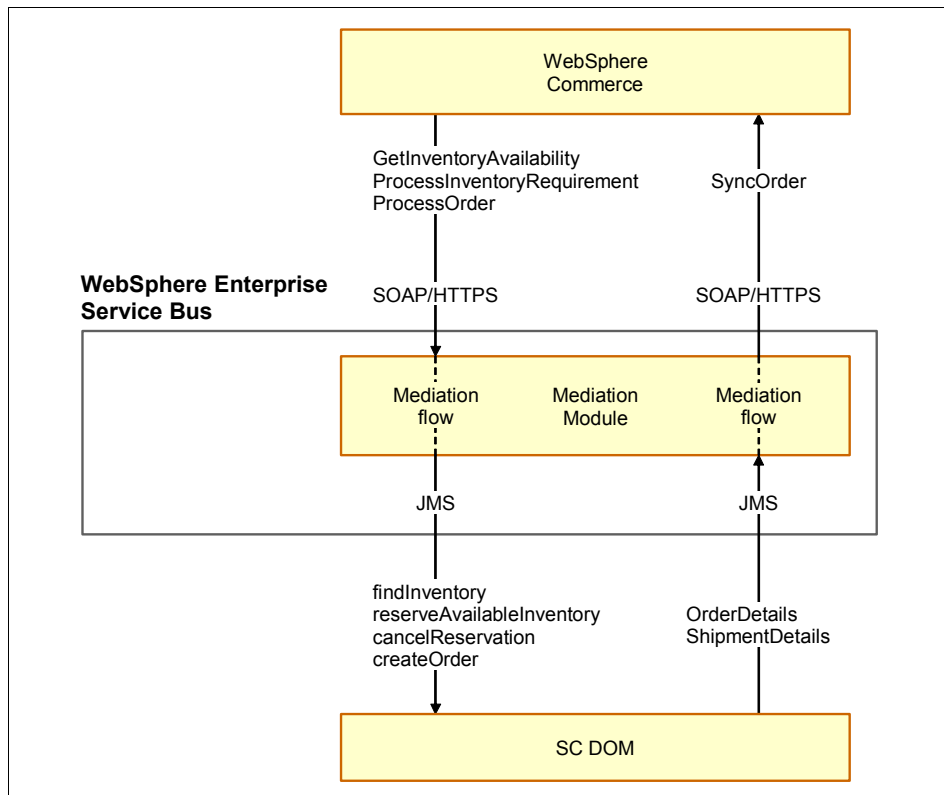


Figure 7-3 Communications model for WebSphere Commerce and SSFS integration

7.1.2 Installation, configuration, and deployment

Integrating WebSphere Commerce and SSFS is not a trivial procedure. It involves three relatively complex tasks, performed using WebSphere Commerce, WESB Mediation Module, and SSFS.

High-level steps that must be completed to get the integration to function properly are:

- ▶ Configure WebSphere Commerce to use an external order management system (OMS).
- ▶ Modify the WESB mediation module so that the data is translated correctly between WebSphere Commerce and SSFS.
- ▶ Deploy the WESB mediation module.
- ▶ Configure SSFS to respond to JMS API requests.
- ▶ Configure SSFS to send order status updates to WebSphere Commerce.
- ▶ Synchronize data between WebSphere Commerce and SSFS.

7.2 WebSphere Commerce installation and configuration

To use SSFS as an external OMS, a few configuration changes must be made to WebSphere Commerce. To enable SSFS integration, WebSphere Commerce must be installed at Version 7.0.0.2 or later. When using WebSphere Commerce 7.0.0.2, Feature Pack 2 (FEP2) must be installed to gain access to the SSFS integration code for WebSphere Enterprise Service Bus.

For a store to be able to use SSFS integration code, the store must be deployed using the external inventory model. This activity is done during the store publish processing the commerce site Administration Console, enabling and pre-populating the inventory caching tables crucial to the integration.

Once the store is published with external inventory, the store can be altered to use DOM. The instructions for enabling DOM integration are available at the following website under the topic “Configuring the DOM integration feature”:

<http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.sterling-integration.doc/concepts/cyasterling.htm>

Important: Use the following URLs for the message types to point to the WESB mediation module:

```
External Inventory System - http://<WESB_HOSTNAME>:<WESB_PORT>/WCToSSFSMediationModuleWeb/sca/WCInventoryServicesExport
External Order System - http://<WESB_HOSTNAME>:<WESB_PORT>/WCToSSFSMediationModuleWeb/sca/WCOrderServicesExport
```

When using the Madison starter store, to enable the updated order status display, copy all sub folders from:

WC(DE)_install\dir\components\sterling-integration\samples\stores\Madisons

To:

WC_eardir\Stores.war\storedir

To further facilitate inventory caching in the store, add the necessary rows to the database tables listed on the following website:

<http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.dom-integration.doc/concepts/csmdominventorycaching.htm>

7.3 WESB mediation module installation and configuration

The method of integration between WebSphere Commerce and SSFS is a WebSphere Enterprise Service Bus (WESB) mediation module. This module requires WebSphere Integration Developer (WID) 7.0.0.3. To extend or modify the mediation module, you must import the project into a WID workspace. WID enables development changes made to the mediation to be compiled and deployed to a WESB server. During development, the WESB server deployed with WID can be used to test integrations.

7.3.1 Importing the mediation module into WID

Here are the steps to importing the mediation module into WID:

1. Open the WebSphere Integration Developer workspace and from the menu click **File** → **Import**.
2. Click **General** → **Existing projects into Workspace** (Figure 7-4).

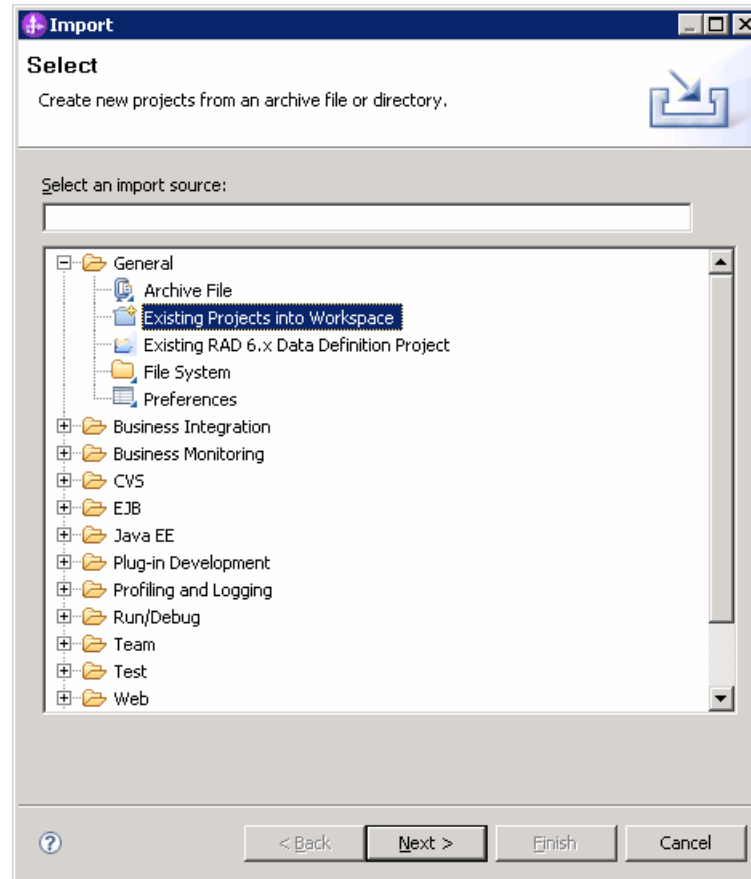


Figure 7-4 WID Select Existing Project pane

3. Select the Mediation Module project archive file (Figure 7-5). The project can be either a JAR or a folder. The mediation module project can be found in `WCDE_installdir/components/sterling-integration/wid` with the name `WCToSSFSMediationModule`.

Note: The mediation module has certain dependencies from WebSphere Commerce. Add the `Foundation-Core.jar` and `Foundation-Server.jar` libraries from the `WC/lib` directory.

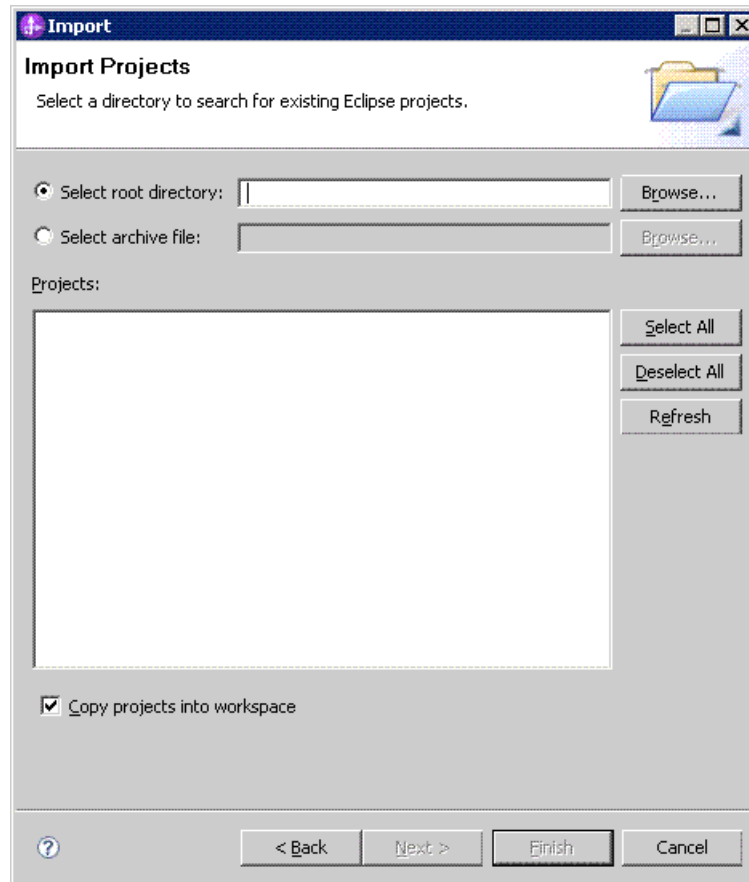


Figure 7-5 WID Import Existing Project pane

7.3.2 Making necessary changes to the mediation module

No two WebSphere Commerce or SSFS installations will be the same. Once the mediation module has been imported into your WID workspace, you can modify it

as necessary so that it will work within your specific environment. With this said, certain changes will always have to be made before it can be used.

Setting WebSphere Commerce Order Services endpoint URL

For the WESB mediation module to communicate correctly with WebSphere Commerce, it must be configured with the proper hostname and port. To accomplish this, perform these steps:

1. Open the assembly diagram in the WCToSSFSMediationModule.
2. Open the **Properties** view.
3. Click **WCOOrderServicesImport**.
4. Choose the **Binding** tab in the properties view.
5. Enter the address of your WebSphere Commerce Order update service (Figure 7-6).

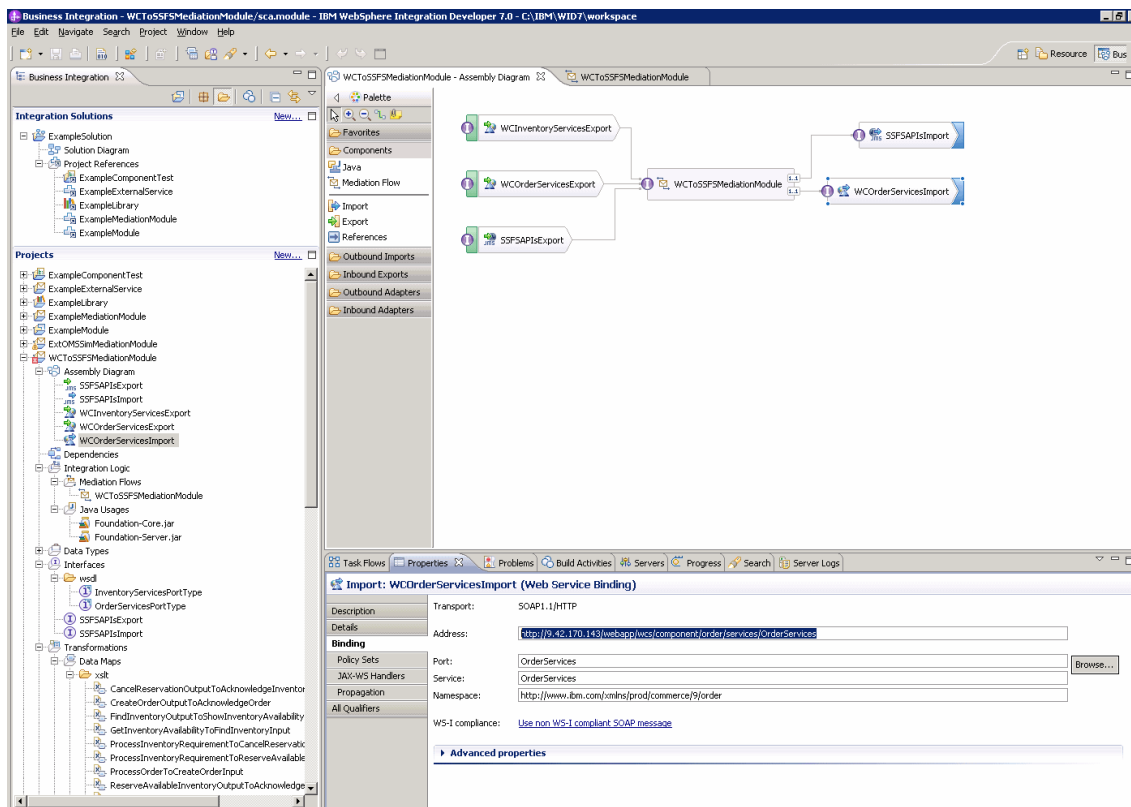


Figure 7-6 Enter the address of the WCOOrderServicesImport in WID window

Note: This service is provided by WebSphere Commerce. The default address for a WebSphere Commerce installation is:

<https://hostname:8000/webapp/wcs/services/OrderServices>

For more information about this topic, see the WebSphere Commerce Infocenter:

<http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.webservices.doc/refs/rwvorderstatusupdateservice.htm>

WebSphere Commerce Web Services Security

WebSphere Commerce can use WS-Security to provide security for its web services. The SSFS integration module for WESB does not utilize this security.

The steps to enable WS-Security are listed below. To enable WS-Security, the WSServicesPolicySet.zip policy file is required, obtained from the WebSphere Commerce Infocenter, at this address:

<http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.sterling-integration.doc/code/WCServicesPolicySet.zip>

The WESB mediation module must be configured to utilize security correctly. Perform these steps:

1. Launch the WESB integrated solutions console and log on.
2. Navigate to **Services** → **Policy set** → **Application policy sets** and select **Import** → **From selected location**.
3. Select **WCServicesPolicySet.zip** and click **OK**.
4. Navigate to **Services** → **Service clients** and select **OrderServices**.
5. Select **OrderServices** and select **Attach Client Policy Set** → **WCServicesPolicySet**.
6. Select **OrderServices** and select **Attach Binding** → **New Application Specific Binding**.
7. Enter **WCServicesBinding** as the name of the binding configuration and select **Add** → **WS-Security**.
8. Navigate to **Authentication and protection** and select **request:token_auth**.
9. Click **Apply**.
10. Navigate to **Callback handler**.
11. Enter the **wcsadmin** username and password and click **OK**.
12. Save the WESB configuration changes.

Adjusting the SSFS organization code in the mediation flows

Most calls to the SSFS APIs require an organization code. As this code is global, specify it in the WESB mediation module. To change this code to fit your environment, perform the following steps:

1. Open the **WCToSSFSMediationModule** component in WID.
2. For each of the services listed below, adjust the value set in the **OrganizationCodeSetter** element:
 - **ProcessOrder**
 - **GetInventoryAvailability**
 - **ProcessInventoryRequirement**

Enabling and disabling mediation flow tracing

Trace primitives are used all over the mediation flows to log messages in the server's `SystemOut.log` (for example, the incoming `GetInventoryAvailability` BOD and the resulting `findInventory` input message). In WID, you can enable and disable these traces by opening the mediation flows. The enablement of these traces are also set as promoted properties with aliases in the form of `*Trace.enabled` (for example, `GetInventoryAvailabilityTrace.enabled` and `FindInventoryInputTrace.enabled`). They can be enabled and disabled in WESB using the Administration Console. See details about promoted properties in the WebSphere Integration Developer InfoCenter:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/topic/com.ibm.wbit.help.mediation.doc/topics/cpromoted.html>

Deploying the mediation module to WESB

After changes have been made and tested using WID and the WESB test environment, you can export the mediation module as a deployable resource. You can then deploy the ear into a standalone WESB install.

To export a deployable resource from WID, perform these steps:

1. Right-click the **WCToSSFSMedationModule** project and click **Export**.
2. Select **Integration Modules and Libraries**.
3. Ensure that **Files for Server Deployment** is selected and click **Next**.
4. Select the location of the generated EAR file and click **Finish**.

To install the generated EAR file into a WESB server, perform the following:

1. Log into the WESB admin console.
2. Navigate to **Applications** → **SCA Modules**.
3. Click **Install**.
4. Click **Browse** and select the mediation module EAR.
5. Select the server or cluster to install the module into and click **Next**.
6. When the install completes, click **Save**.

See the WebSphere Integration Developer Infocenter and the WESB Infocenter for more information:

- WebSphere Integration Developer Infocenter

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/index.jsp?topic=/com.ibm.wbit.help.mainwelcome.doc/topics/welcome.html>

- WebSphere Enterprise Service Bus Infocenter

http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/index.jsp?topic=/com.ibm.websphere.wesb.doc/doc/welcome_wps_ovw.html

Configuring JMS for WESB

The SSFS integration module configures WESB to communicate with SSFS via JMS. The environment must be configured to enable such communications. The SSFS uses the default messaging implementation of WESB server as the JMS provider. When the WESB mediation module is deployed to the server, most of the necessary configuration is completed automatically. After the module is deployed, the only thing required is to create a JMS Queue Connection factory.

To create a JMS Queue Connection factory, perform these steps:

1. Log into the WESB admin console.
2. Navigate to **Resources** → **JMS** → **Queue Connection Factories**.
3. Click **New**.
4. Choose **Default Messaging Provider** and click **Next**.
5. Enter the following required information and click **OK**.

Name - WCToSSFSMediationModule.SSFSAPIsImport_QCF
JNDI Name - WCToSSFSMediationModule/SSFSAPIsImport_QCF
Bus Name - SCA.APPLICATION.<CELLNAME>.Bus
Bus Provider Endpoints - <SERVER
HOSTNAME>:7276:BootstrapBasicMessaging, <SERVER
HOSTNAME>:5557:BootstrapSecureMessaging

6. Click **OK**, then click **Save**.

7.4 Installing, configuring, and deploying SSFS

SSFS is a very flexible and powerful system. Due to this, each installation is configured differently. This section includes common configurations that must be made to SSFS for it to work with the mediation module.

The SSFS installation (Version 9.0) created during the writing of this book was deployed on WebSphere Application Server v7.0.0.11 on 64-bit Red Hat Enterprise Linux®. The SSFS installation used the same DB2 database server as the WebSphere Commerce instance, but with a separate tablespace and schema.

For a general overview of how WebSphere Commerce fields map to SSFS fields, see the general mapping rules at this address:

<http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.sterling-integration.doc/refs/ryafLOWgeneral.htm>

7.4.1 Enabling inbound API calls over JMS

To enable SSFS to receive API calls over JMS, you need to create a custom async service for each API call. The SSFS API calls that we need to enable are:

- ▶ findInventory
- ▶ reserveAvailableInventory
- ▶ createOrder

To enable an API call, we create a new service that:

- ▶ Receives the request on a JMS queue
- ▶ Performs the API call
- ▶ Puts the response on another JMS queue

Figure 7-7 shows the standard inbound flow.

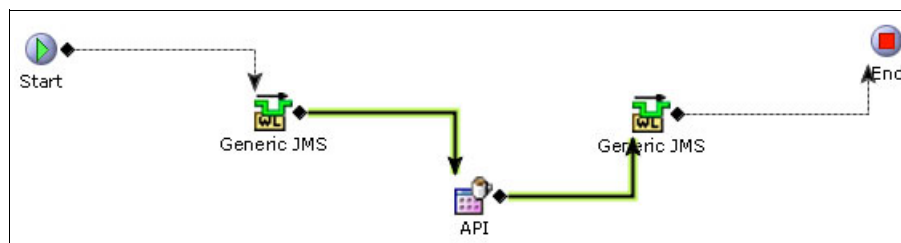


Figure 7-7 Service definition of an SSFS

To create an API service, perform the following steps:

1. Launch the **Applications Manager**.
2. Select **Applications** → **Applications Platform** from the menu.
3. Double-click **Process Modeling**.
4. Double-click the Order Fulfillment icon.
5. Click the **Service Definitions** tab at the bottom of the left pane.
6. Create a new service called EXTN_WC_<API NAME>, where <API NAME> is the name of the API that you will be calling. Ensure that it is added to the WebSphere Commerce group.
7. Drag components onto the palette so that it resembles the diagram shown in Figure 7-7 on page 140.
8. For the line extending from the first generic JMS node to the API node, use the following steps for the configuration parameters.

Note: These are just sample parameters. Your environment might be different, so some of these might have to change. In particular, you will have to update the provider URL to match the hostname and BOOTSTRAP_ADDRESS of your WESB server.

If a parameter does not appear in the tables below, it denotes that the field is left blank.

- a. Set the Runtime tab configuration parameters for SSFS as given in Table 7-1.

Table 7-1 Runtime tab configuration parameters of SSFS

Parameter	Value
Sub Service Name	WC_findInventory
Queue Name	WCToSSFSMediationModule/SSFSAPIsImport_SEND_D
Provider URL	corbaloc:iiop:<WESB server name or IP>:<WESB server BOOTSTRAP_ADDRESS>
Initial Context Factory	WebSphere MQ
QCF Lookup	WCToSSFMediationModule/SSFSAPIsImport_QCF
Transactional	Selected

Parameter	Value
Initial Threads	5
Selector	TargetFunctionName = 'findInventory'
Default Reply To Queue Name	WCToSSFSMediationModule/SSFSAPISImport_RECEIVE_D

The Runtime tab SSFS should look similar to Figure 7-8 after inserting the data according to the parameter values in Table 7-1 on page 141.

Properties: JMS Receiver

Runtime Server Reconnect Exception Exception References Jms Security Properties

Sub Service Name WC_findInventory Queue Name WCToSSFSMediationModule/SSFSAPISImport_SEND_D

Provider URL corbaloc:iiop:terminus.torolab.ibm.com:9104 Initial Context Factory WebSphere MQ

QCF Lookup WCToSSFSMediationModule/SSFSAPISImport_QCF

Initial Threads 5

Service To Execute On EOF Message

Root Node Name Of EOF Message

Default Reply To Queue Name WCToSSFSMediationModule/SSFSAPISImport_RECEIVE_D

Enable JMS Security Process Reply To Queue

Transactional Non Transactional

Figure 7-8 Runtime tab configuration parameters

- b. Set the Server tab configuration parameters for SSFS as given in Table 7-2.

Note: The server is used to identify a SSFS integration server. SSFS integration servers run separate from the SSFS main instance and are usually started by the ./startIntegrationServer.sh script in the SSFS/bin directory.

Table 7-2 Server tab configuration parameters of SSFS

Parameter	Value
Server Name	WC_API_Receiver
Needs Decompression	Not selected

For the Server tab, SSFS should look similar to Figure 7-9 after inserting the parameter values found in Table 7-2 on page 142.



Figure 7-9 Server tab configuration parameters view of SSFS

- c. Set the Reconnect tab configuration parameters for SSFS using Table 7-3.

Table 7-3 Reconnect tab configuration parameters of SSFS

Parameter	Value
Maximum Time Between Reconnects (min)	2

The Reconnect tab for SSFS should look similar to Figure 7-10 after inserting the parameter values given in Table 7-3.

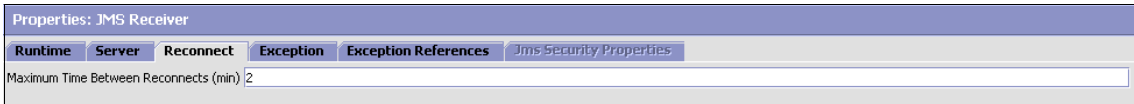


Figure 7-10 Reconnect tab configuration parameters view of SSFS

- d. Set the Exception tab configuration parameters as given in Table 7-4.

Table 7-4 Exception tab configuration parameters of SSFS

Parameter	Value
Alert Type	WC_API_Exception
Alert Queue Name	System Announcement Queue (DEFAULT)
Suspend API	Not selected
Is Reprocessible	Not selected

The Reconnect tab for SSFS should look similar to Figure 7-11 after inserting the parameter values from Table 7-4 on page 143.

Properties: JMS Receiver

Runtime Server Reconnect Exception Exception References Jms Security Properties

Alert Type: WC_API_Exception Alert Queue Name: System Announcement Queue (DEFAULT)

☐ Suspend API Suspend Wait Time (seconds):

☐ Is Reprocessable ☐ Check for Prior Exception

Exception Group: Prior Errors User Exit:

Figure 7-11 Exception tab configuration parameters view of SSFS

9. For the line extending from the API node to the second generic JMS nodes, set the Runtime tab configuration parameters for SSFS as in Table 7-5.

Table 7-5 Runtime tab configuration parameters of SSFS

Parameter	Value
Queue Name	WCToSSFSMediationModule/SSFSA PIsImport_RECEIVE_D
Time To Live (seconds)	10
Provider URL	corbaloc:iiop:terminus.torolab.ibm.co m:9104
Initial Context Factory	WebSphere MQ
QCF Lookup	WCToSSFSMediationModule/SSDS APIsImport_QCF
Persistent	Selected
Needs Compression	Not selected
Commit of this message depends on parent transaction	Selected
Enable JMS Security	Not selected

The Reconnect tab for SSFS should look similar to Figure 7-12 after inserting the parameter values found in Table 7-5 on page 144.

Figure 7-12 Runtime tab configuration parameters view of SSFS

7.4.2 Enabling outbound API calls over JMS

For the outbound API call to operate, we need to:

- ▶ Create a service definition for the API.
- ▶ Link the service definition to an action.
- ▶ Link the action to a transaction.
- ▶ Trigger the transaction by an agent service.

Creating the outbound service definitions

Similarly to enabling inbound API calls, for SSFS to notify WebSphere Commerce of changes to the order status, we must set up custom services. These services will:

- ▶ Make an API call.
- ▶ Send the results of that call over a JMS queue.

The flow appear as in Figure 7-13.

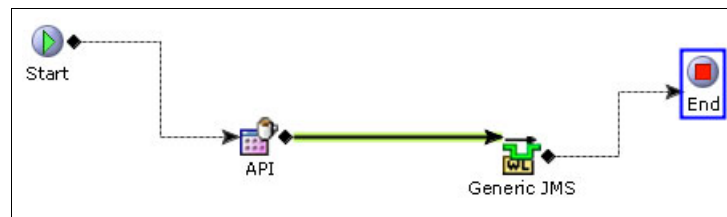


Figure 7-13 Service definition of an SSFS

After we create the services, they can be added to an action that, in turn, can be added to a transaction event. When the transaction event is fired, it calls the action, which calls the service. The service then sends order or shipment details to WebSphere Commerce.

We next create two custom services. The first notifies WebSphere Commerce of order updates and the second updates WebSphere Commerce of shipment updates.

Creating the send order details outbound service definition

To create the send order details service, perform the following steps:

1. Launch the **Applications Manager**.
2. Select **Applications** → **Applications Platform** from the menu.
3. Double-click **Process Modeling**.
4. Double-click the Order Fulfillment icon.
5. Click the **Service Definitions** tab at the bottom of the left pane.
6. Create a new service called EXTN_WC_sendOrderDetails_JMS. Ensure that **In a Synchronous Mode** is selected. Ensure that it is added to the WebSphere Commerce group.
7. Drag the components onto the palette so that it resembles Figure 7-13 on page 145.
8. Double-click the API node and set the API name parameter to getOrderDetails (Figure 7-14).

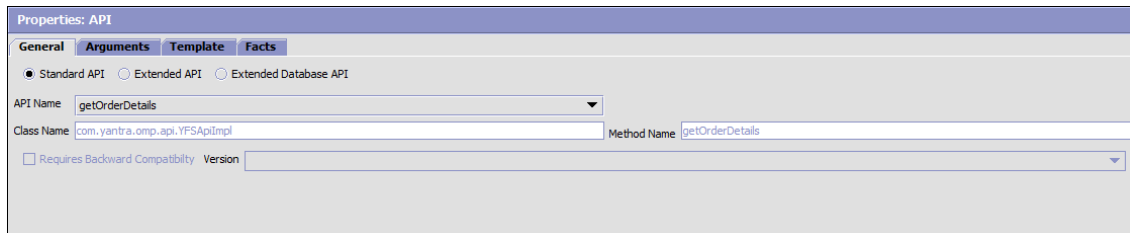


Figure 7-14 API Parameter General pane

9. For line extending from the first API node to the generic JMS node, use the following for the configuration parameters.

Note: These are sample parameters. Your environment might be different, so some parameters might have to change. In particular, you will have to update the provider URL to match the hostname and BOOTSTRAP_ADDRESS of your WESB server.

If a parameter does not appear in the tables below, it denotes that the field is left blank.

- a. Set the Runtime tab configuration parameters of SSFS using Table 7-6.

Table 7-6 Runtime tab configuration parameters of SSFS

Parameter	Value
Queue Name	WCToSSFSMediationModule/SSFSAPIsExport_RECEIVE_D
Provider URL	corbaloc:iiop:your_host_name:2809
Initial Context Factory	WebSphere MQ
QCF Lookup	WCToSSFSMediationModule/SSFSAPIsImport_QCF
Persistent	Selected
Needs Compression	Not selected
Commit of this message depends on parent transaction	Selected
Enable JMS Security	Not selected

The Runtime tab SSFS should look similar to Figure 7-15 after inserting the parameter values found in Table 7-6.

Properties: JMS Sender

Runtime Header Reconnect Jms Security Properties

Queue Name: WCToSSFSMediationModule/SSFSAPIsExport_RECEIVE_D Time To Live (seconds): 10

Provider URL: corbaloc:iiop:hitech.torolab.ibm.com:2809 Initial Context Factory: WebSphere MQ

QCF Lookup: WCToSSFSMediationModule/SSFSAPIsImport_QCF

☒ Persistent ☐ Non Persistent

☐ Needs Compression ☒ Commit of this message depends on parent transaction

☐ Enable JMS Security

Figure 7-15 Runtime tab configuration parameters view of SSFS

- b. Set the Header tab configuration parameters using Table 7-7.

Table 7-7 Header tab configuration parameters

Parameter	Value
TargetFunctionName	sendOrderDetails

The Header tab SSFS should look similar to Figure 7-16 after inserting the parameter values found in Table 7-7.

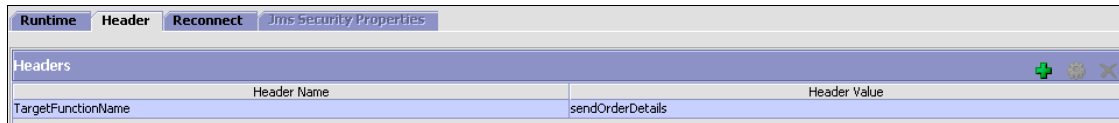


Figure 7-16 Header tab configuration parameters view of SSFS

- c. Set the Reconnect tab configuration parameters for SSFS using Table 7-8.

Table 7-8 Reconnect tab configuration parameters of SSFS

Parameter	Value
Retry Interval (ms)	0
Number of Retries	0
Use Backup JMS	Not selected

The Reconnect tab SSFS should look similar to Figure 7-17 after inserting the parameter values found in Table 7-8.

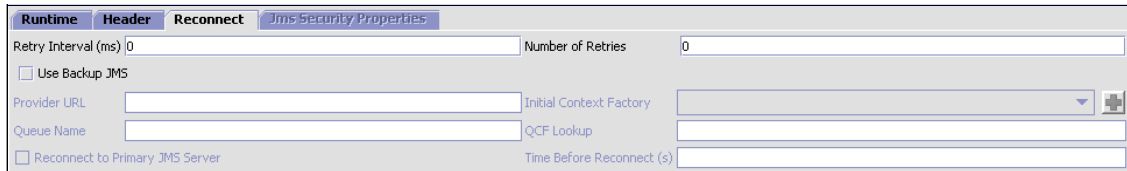


Figure 7-17 Reconnect tab configuration parameters view of SSFS

Creating the send order details outbound service definition

To create the send order details service, perform the following steps:

1. Launch the **Applications Manager**.
2. Select **Applications** → **Applications Platform** from the menu.
3. Double-click **Process Modeling**.

4. Double-click the Outbound Shipment icon.
5. Click the **Service Definitions** tab at the bottom of the left pane.
6. Create a new service called EXTN_WC_sendShipmentDetails_JMS. Ensure that it is added to the WebSphere Commerce group.
7. Drag components onto the palette so that it resembles Figure 7-13 on page 145.
8. Double-click the API node and set the API Name parameter to **getShipmentDetails**.
9. For the line extending from the first API node to the generic JMS node, use the configuration parameters in Table 7-9. (Note that these are sample parameters. Your environment may be different so some of them may have to change.)
 - a. Set the Runtime tab configurations as shown in Table 7-9.

Table 7-9 Runtime tab configuration parameters of SSFS

Parameter	Value
Queue Name	WCToSSFSMediationModule/SSFSAPIsExport_RECEIVE_D
Time To Live (seconds)	10
Provider URL	corbaloc:iiop:your_host_name:2809
Initial Context Factory	WebSphere MQ
QCF Lookup	WCToSSFSMediationModule/SSFSAPIsImport_QCF
Persistent	Selected
Needs Compression	Not selected
Commit of this message depends on parent transaction	Selected
Enable JMS Security	Not selected

The Runtime tab SSFS should look similar to Figure 7-18 after inserting the parameter values from Table 7-9 on page 149.

Properties: JMS Sender

Runtime Header Reconnect Jms Security Properties

Queue Name: WCToSSFSMediationModule/SSFSAPISExport_RECEIVE_D Time To Live (seconds): 10

Provider URL: corbaloc:iiop:hitech.torolab.ibm.com:2809 Initial Context Factory: WebSphere MQ

QCF Lookup: WCToSSFSMediationModule/SSFSAPISImport_QCF

☐ Persistent ☐ Non Persistent

☐ Needs Compression ☒ Commit of this message depends on parent transaction

☐ Enable JMS Security

Figure 7-18 Runtime tab configuration parameters view of SSFS

- b. Set the Header tab configuration parameters using Table 7-10.

Table 7-10 Header tab configuration parameters of SSFS

Parameter	Value
TargetFunctionName	sendShipmentDetails

The Header tab SSFS should look similar to Figure 7-19 after inserting the parameter values from Table 7-10.

Properties: JMS Sender

Runtime Header Reconnect Jms Security Properties

Headers

Header Name	Header Value
TargetFunctionName	sendShipmentDetails

Figure 7-19 Header tab configuration parameters view of SSFS

- c. Set Reconnect tab configuration parameters using Table 7-11.

Table 7-11 Reconnect tab configuration parameters of SSFS

Parameter	Value
Retry Interval (ms)	0
Number of Retries	0
Use Backup JMS	Not selected

The Reconnect tab SSFS should look similar to Figure 7-20 after inserting the the parameter values from Table 7-11 on page 150.

Figure 7-20 Reconnect tab configuration parameters view of SSFS

Linking send order details service definition to actions

To create an action for the EXTN_WC_sendOrderDetails_JMS service definition:

1. Click the **Actions** tab at the bottom of the left-hand pane.
2. Click the plus icon (+) at the top of the left pane to add a new action.
3. Enter the following information to create the base action definition:
 - Action Code: SendOrderDetails
 - Action Name: Send Order Details
 - Action Group: WebSphere Commerce

Then:

- a. Ensure that **Invoke the following services as part of this action** is checked.
- b. To the list of Invoked Services, add the Service Definition name: EXTN_WC_sendOrderDetails_JMS
4. Click **Save**.

Linking the send order details action to transactions

The send order details action must be linked to three transactions:

- ▶ Release order: ON_SUCCESS
- ▶ Release order: ON_RELEASE_CREATION_OR_CHANGE
- ▶ Schedule order: ON_BACKORDER

To link the send order details action to the three transactions:

1. Go to the **Transactions** tab at the bottom of the left-hand pane.
2. Double-click the **Release Order** transaction.
3. Double-click the **ON_SUCCESS** event.
4. Drag the send order details action from the Actions tab of the left pane to the diagram in the right pane.

5. Right-click the **ON_SUCCESS** event.
6. Open the **Details** pane.
7. Check the **Is Active?** check box.
8. Click **OK**.
9. Double-click the **ON_RELEASE_CREATION_OR_CHANGE** event.
10. Drag the send order details action from the Actions tab of the left pane to the diagram in the right pane.
11. Right-click the **ON_RELEASE_CREATION_OR_CHANGE** event.
12. Open the **Details** pane.
13. Check the **Is Active?** check box.
14. Click **OK**.
15. Click the **Time Triggered** tab in the transaction window.
16. Ensure that the **Transaction is Time Triggered (an agent)** check box is checked.
17. Add an **Agent Criteria Definition**.
18. Enter the following parameters:
 - Criteria ID: SCHEDULE.0001
 - Agent Server: WebSphereAgent
 - JMS Queue Name:
WCToSSFSMediationModule/SSFSAPISAgent_sendOrderDetails
 - No. of Threads: 1
 - Initial Context Factory: WebSphere MQ
 - QCF Lookup: WCToSSFSMediationModule/SSFSAPISImport_QCF
 - Provider URL: corbaloc:iiop:<WESB hostname>:<BOOTSTRAP_PORT>
19. Save the **Agent Criteria Details**.
20. Save the transaction.
21. Double-click the **Schedule Order** transaction.
22. Double-click the **ON_BACKORDER** event.
23. Drag the send order details action from the Actions tab on the left pane to the diagram in the right pane.
24. Right-click the **ON_BACKORDER** event.
25. Open the **Details** pane.
26. Check the **Is Active?** check box.

27. Click **OK**.
28. Click the **Time Triggered** tab in the transaction window.
29. Ensure the **Transaction is Time Triggered (an agent)** check box is checked.
30. Add an **Agent Criteria Definition**.
31. Enter the following parameters:
 - Criteria ID: SCHEDULE.0002
 - Agent Server: WebSphereAgent
 - JMS Queue Name:
WCToSSFSMediationModule/SSFSAPISAgent_sendOrderDetails
 - No. of Threads: 1
 - Initial Context Factory: WebSphere MQ
 - QCF Lookup: WCToSSFSMediationModule/SSFSAPISImport_QCF
 - Provider URL: corbaloc:iiop:<WESB hostname>:<BOOTSTRAP_PORT>
32. Save the **Agent Criteria Details**.
33. Save the **Transaction**.

Linking send shipment details service definition to an action

To create an action for the EXTN_WC_sendShipmentDetails_JMS Service definition:

1. Click the **Actions** tab at the bottom of the left-hand pane.
2. Click plus icon (+) at the top of the left pane to add a new action.
3. Enter the following information to create the base action definition:
 - Action Code: SendShipmentDetails
 - Action Name: Send Shipment Details
 - Action Group: WebSphere Commerce
 - a. Ensure that **Invoke the following services as part of this action** is checked.
 - b. For Invoked Services enter EXTN_WC_sendShipmentDetails_JMS.
4. Click **Save**.

Linking the send shipment details action to a transaction

The send shipment details action must be linked to the confirm shipment: ON_SUCCESS transaction.

To link the send shipment details action to the confirm shipment: ON_SUCCESS transaction

1. Go to the **Transactions** tab at the bottom of the lefthand pane.
2. Double-click the **Confirm Shipment** transaction.
3. Double-click the **ON_SUCCESS** event.
4. Drag the send shipment details action from the Actions tab on the left pane to the diagram in the right pane.
5. Right-click the **ON_SUCCESS** event.
6. Open the **Details** pane.
7. Check the **Is Active?** check box.
8. Click **OK**.
9. Save the **Agent Criteria Details**.
10. Save the transaction.

After these services are created, they can be added to actions, which can be added to transactions. A good example of transactions to use are release order and shipment shipped.

Adding send order details JMS Queue to WebSphere Enterprise Service Bus

During the enablement of the Sterling transactions, agents were configured to use WebSphere JMS queues. These queues are not created automatically. Therefore, they must be configured.

The queues are used by SSFS to send transaction notifications to the agent. When the agent receives the notification, it invokes the SSFS API to retrieve the required information and transmits this information to the specified destination.

To configure the WCToSSFSMediationModule/SSFSAPIsAgent_sendOrderDetails queue, perform the following steps:

1. Log into the WebSphere Integration Solutions Console.
2. Create the destination queue WCToSSFSMediationModule.SSFSAPIsAgent_sendOrderDetails_SIB.
 - a. Select **Service integration** → **Buses**.
 - b. Select the **SCA.APPLICATION.qcell.Bus** bus.
 - c. Select **Destinations** in the Destination resources section.
 - d. Select **New** to create a new resource.

- e. Select **Queue** and click **Next** to specify a new queue.
 - f. Set the identifier and description to **WCToSSFSMediationModule.SSFSAPIsAgent_sendOrderDetails_SIB** and click **Next**.
 - g. Click **Next** to assign the queue to the default bus member.
 - h. Click **Finish** to complete the queue definition.
 - i. Save to the master configuration.
3. Create the JMS queue WCToSSFSMediationModule/SSFSAPIsAgent_sendOrderDetails by performing these steps:
 - a. Select **Resources** → **JMS** → **Queues**.
 - b. Select **New** to create a new queue.
 - c. Click **OK** to accept the default messaging provider.
 - d. Set Name to **WCToSSFSMediationModule.SSFSAPIsAgent_sendOrderDetails**.
 - e. Set JNDI name to **WCToSSFSMediationModule/SSFSAPIsAgent_sendOrderDetails**.
 - f. Select Bus **SCA.APPLICATION.qcell.Bus**.
 - g. Select Queue **WCToSSFSMediationModule.SSFSAPIsAgent_sendOrderDetails_SIB**.
 - h. Click **OK** to create the JMS queue.
 - i. Save to the master configuration.
 4. Log out of the WebSphere Integration Solutions Console.

Updating Sterling Agent Classpath to use WebSphere JMS

To enable SSFS agents to send and receive WebSphere JMS communication, libraries must be added to the agent classpath. The library list consists of:

- ▶ <WAS base>/runtimes/com.ibm.ws.sib.client.thin.jms_7.0.0.jar
- ▶ <WAS base>/runtimes/com.ibm.ws.ejb.thinclient_7.0.0.jar
- ▶ <WAS base>/runtimes/com.ibm.ws.orb_7.0.0.jar
- ▶ <WAS base>/plugins/com.ibm.ws.wccm.jar
- ▶ <WAS base>/plugins/com.ibm.ws.runtime.jar
- ▶ <WAS base>/plugins/com.ibm.ws.security.crypto.jar

For example, on a UNIX® environment where the Sterling base directory is /opt/IBM/Sterling/Foundation/ and the WebSphere Application Server base

directory is /opt/IBM/WebSphere/AppServer, the commands are shown in Example 7-1.

Example 7-1 Commands for registering agent libraries for JMS

```
cd /opt/IBM/Sterling/Foundation/bin
./install3rdParty.sh WebSphere 7_0_0_7 -j
/opt/IBM/WebSphere/AppServer/runtimes/com.ibm.ws.sib.client.thin.jms_7.
0.0.jar -targetJVM AGENT
./install3rdParty.sh WebSphere 7_0_0_7 -j
/opt/IBM/WebSphere/AppServer/runtimes/com.ibm.ws.ejb.thinclient_7.0.0.j
ar -targetJVM AGENT
./install3rdParty.sh WebSphere 7_0_0_7 -j
/opt/IBM/WebSphere/AppServer/runtimes/com.ibm.ws.orb_7.0.0.jar
-targetJVM AGENT
./install3rdParty.sh WebSphere 7_0_0_7 -j
/opt/IBM/WebSphere/AppServer/plugins/com.ibm.ws.wccm.jar -targetJVM
AGENT
./install3rdParty.sh WebSphere 7_0_0_7 -j
/opt/IBM/WebSphere/AppServer/plugins/com.ibm.ws.runtime.jar -targetJVM
AGENT
./install3rdParty.sh WebSphere 7_0_0_7 -j
/opt/IBM/WebSphere/AppServer/plugins/com.ibm.ws.security.crypto.jar
-targetJVM AGENT
```

Starting a Sterling Integration Server to process inbound JMS requests

To enable SSFS to receive API calls over JMS, an integration server must be created. The integration server actively monitors the inbound JMS queues and processes any outstanding requests.

Example 7-2 was used when creating this publication to launch the Sterling Integration Server to act on incoming calls to SSFS.

Example 7-2 startIntegrationServer script

```
#!/bin/bash

export STERLING_BASE=/opt/IBM/Sterling/Foundation
export INTEGRATION_LOG=$STERLING_BASE/logs/WC_API_Reciever.log

rm $INTEGRATION_LOG

cd $STERLING_BASE/bin
```

```
nohup ./startIntegrationServer.sh WC_API_Reciever > $INTEGRATION_LOG  
2>&1 &
```

Note: This is example code. For a production instance a service would be set up with stop, start, and restart facilities. Also note that the integration server name matches the server name set on the Sterling Inbound API calls.

Starting a Sterling Agent Server to process outbound JMS requests

To enable SSFS to send API calls over JMS, an agent must be created. The agent actively monitors Sterling and sends any outbound JMS messages as required.

Example 7-3 shows the example code used when creating this publication to launch the Sterling Agent to transmit on outgoing Sterling calls.

Example 7-3 startAgentServer script

```
#!/bin/bash  
  
export STERLING_BASE=/opt/IBM/Sterling/Foundation  
export INTEGRATION_LOG=$STERLING_BASE/logs/WC_API_Sender.log  
  
rm $INTEGRATION_LOG  
  
cd $STERLING_BASE/bin  
nohup ./agentserver.sh WebSphereAgent > $INTEGRATION_LOG 2>&1 &
```

Note: This is example code. For a production instance, a service would be set up with stop, start, and restart facilities. Also note that the integration server name matches the server name set on the Sterling Inbound API calls.

7.5 Configuring SSFS for integration with WebSphere Commerce

This section discusses the admin configuration activities for setting up SSFS for integration. Then this section provides the steps for creating an organization structure and catalog management with an inventory system and apply payment processing-related configurations in the SSFS system similar to WebSphere Commerce so that during catalog browsing from the WebSphere Commerce store the inventory for an item is displayed from the SSFS system.

The main functions that help perform the activities are:

- ▶ Configuring participants
- ▶ Catalog management
- ▶ Global inventory visibility
- ▶ Distribute order management

7.5.1 Configuring participants

Trading partners using the Selling and Fulfillment Foundation to perform supply chain collaborative commerce are called *participants*. Each participant is considered an organization with a defined role. This section covers the creation and edit of an organization.

Creating an organization

Start the Sterling Selling and Fulfillment Suite Application Console (<http://<hostname>/smcfs/console/login.jsp>).

1. Start the Sterling Selling and Fulfillment Suite: Applications Manager.

Click **Launch Application Manager** from Configuration menu (Figure 7-21).

Note: This works only on an Internet Explorer browser. Java must be installed on the local system. Clicking the link prompts you to install Java. Click **yes** to install Java and open the Application Manager.

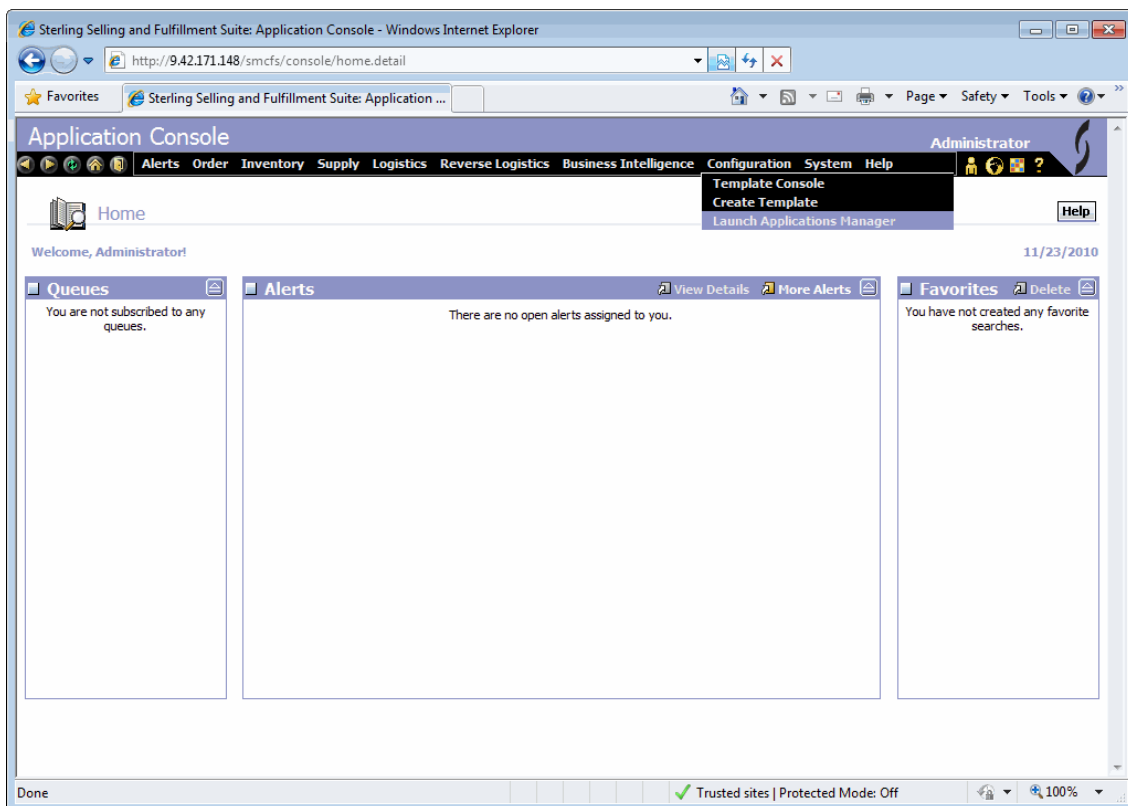


Figure 7-21 Application Console window

2. Click **Applications** → **Application Platform**.

3. Click **Participant Setup** from the tree in the application rules side panel and select **Participant Modeling** → **Participant Setup** (Figure 7-22 on page 160). The Organization Search window displays in the work area.

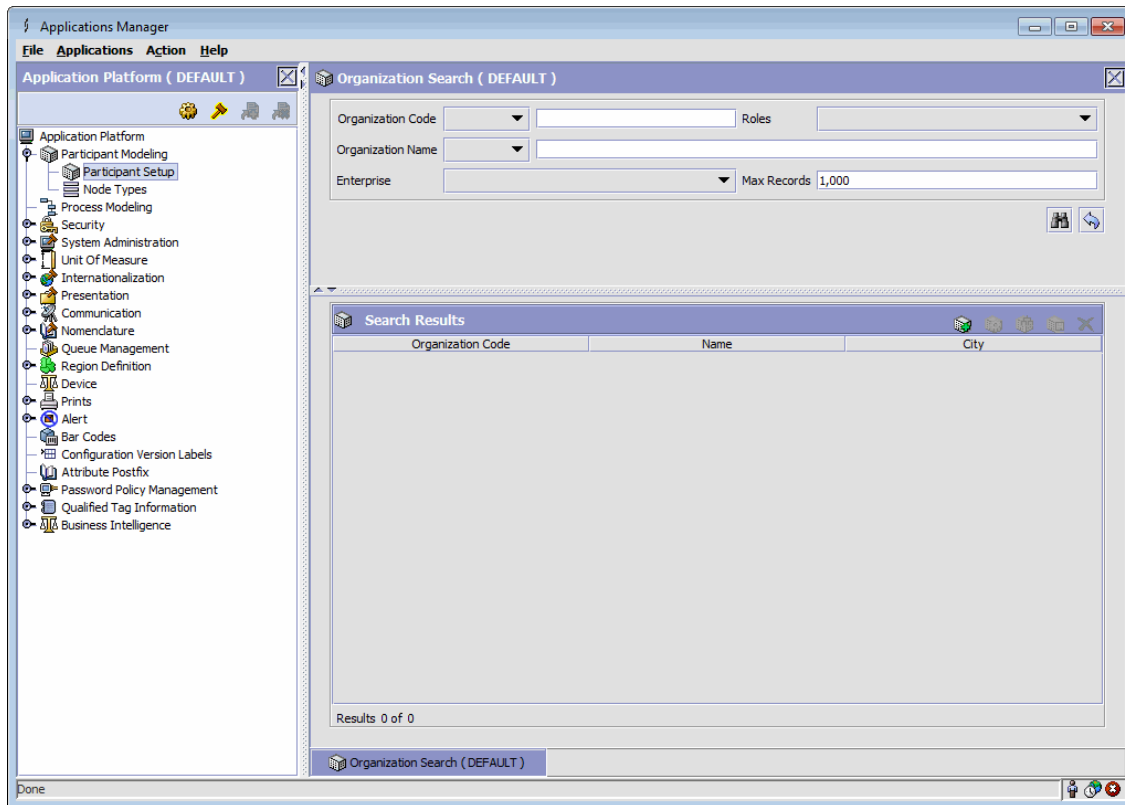


Figure 7-22 Application Manager Participant Setup

4. Click **Create Organization** and the Create Organization pop-up window displays (Figure 7-23 on page 161).

Enter the following detail values:

- Organization Code: MadisonsBase (The default integration uses MadisonsRoot.)

Note: The organization code must not be the same name as the base store or any of the fulfillment centers.

- Organization Name: MadisonsBase

Select the **Organization is an enterprise** check box (Figure 7-23 on page 161). This check box is to configure the organization as an Enterprise.

The screenshot shows the 'Create Organization' pane in the Applications Manager. The form contains the following fields and options:

- Organization Code: Text box with 'MadisonsBase' entered.
- Organization Name: Text box with 'MadisonsBase' entered.
- DUNS Number: Empty text box.
- Account Number With Hub: Empty text box.
- Locale: Dropdown menu with 'US Eastern time' selected.
- Parent Organization: Empty dropdown menu.
- ☒ Organization Is An Enterprise
- Primary Enterprise: Disabled dropdown menu.
- ☐ Organization is a Fulfillment Node
- Node Type: Empty dropdown menu.
- ☐ Organization is a Yard

At the bottom, there are two tabs: 'Address' (selected) and 'Contact Info'. The 'Address' tab is active, showing a large empty text area for the address.

Figure 7-23 Application Manager Create Organization pane

Note: The Primary Enterprise drop-down list is disabled when you select this check box.

Click **Save**.

Note: In the SSFS product documentation is a PDF titled `Application_Platform_Configuration_Guide.pdf`. In section 3.1, “Creating and Modifying an Organization,” Table 3.1 has all the field value descriptions for the Create Organization window.

Editing organization

To edit the newly created organization, double-click the organization and open the organization.

For an organization to function as desired, it must be given one or more roles. Each organization is assigned at least one role. A role is a well-defined set of activities that can be performed by an organization. Each organization performs at least one role. The Selling and Fulfillment Foundation supports buyer, carrier, enterprise, hub, code, and seller organization roles:

1. In the Organization Details window, choose **Roles & Participation** (Figure 7-24).

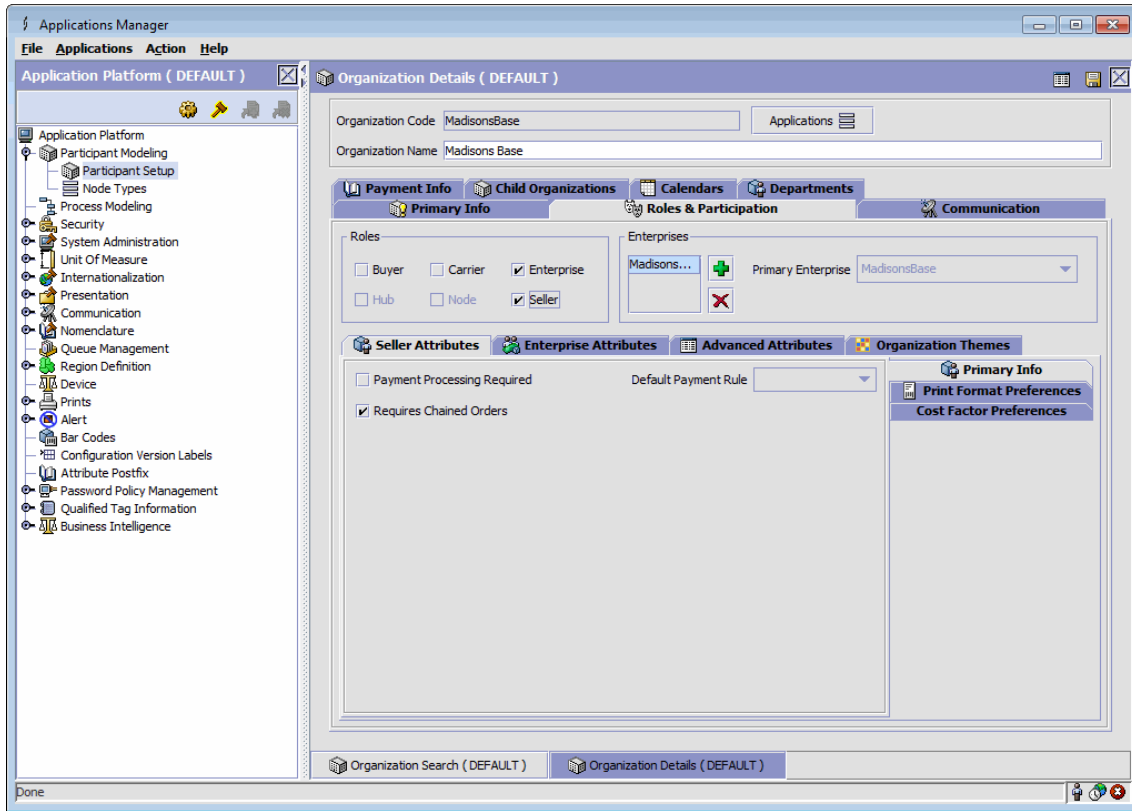


Figure 7-24 Application manager Organization Details Set Seller Role window

2. In the Roles box we have **Enterprise** already selected. Select the role **Seller** too. This adds a new tab in the Seller Attributes window.

Note: An organization is assigned the seller role when it sells product to the Enterprise or other buyer organizations. Sellers can configure payment types, payment rules, and pricing for their organization. When processing orders, a seller organization can use the order, planned order, and purchase order process-type pipelines. A seller organization can only see orders for which it is the buyer, seller, or enterprise. See section 3.1.2, “Assigning the Organization’s Roles and Participant Associations” in SSFS product documentation file [Application_Platform_Configuration_Guide.pdf](#) for more details on organization’s roles and participant associations.

You can determine whether an organization’s inventory is maintained within the Selling and Fulfillment Foundation. You can also determine whether the

organization that you are configuring is an inventory organization or the inventory organization to which it belongs.

3. From the Roles & Participation tab in the Organization Details window, choose **Advanced Attributes** and select the **This Organization Is An Inventory Organization** option (Figure 7-25).

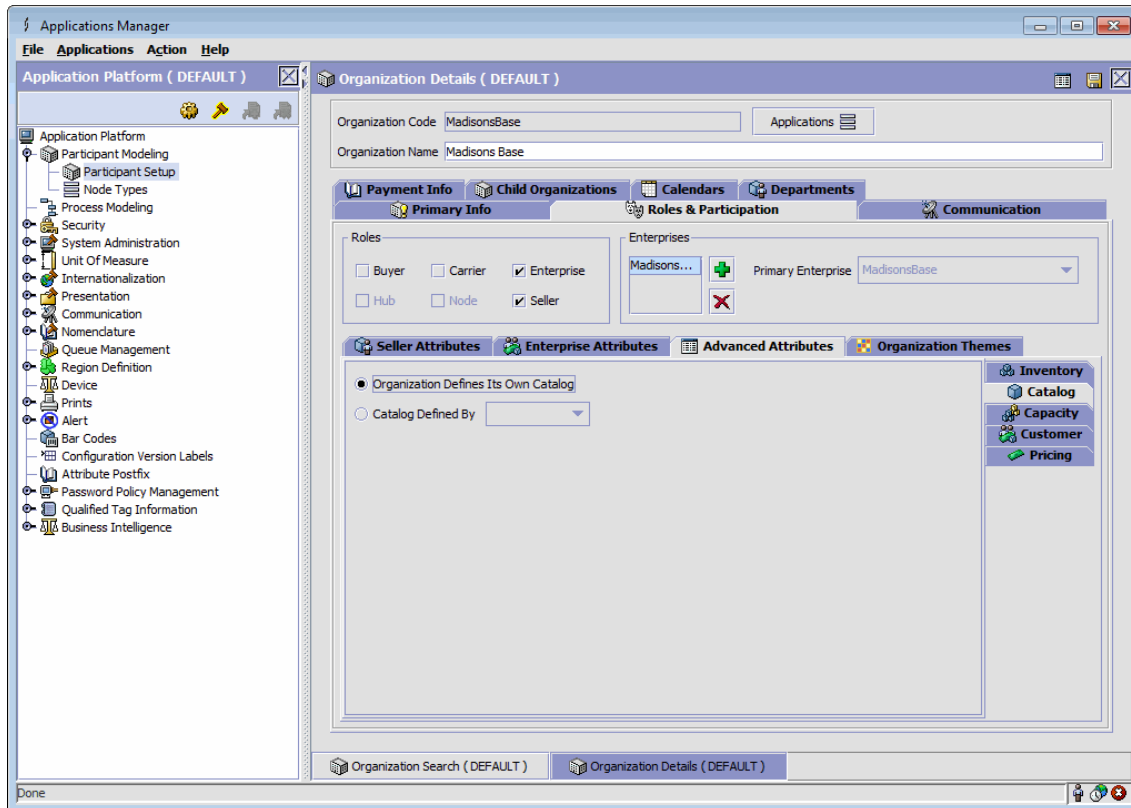


Figure 7-25 Application Manager Organization Details Advance Attribute Catalog Section window

Important: This action is for installation-level configuration only. Do not attempt to reconfigure the parameters on this tab mid-implementation.

Note: When creating an organization through the save as operation, the new organization's catalog organization is the catalog organization of the source organization. If the source organization is its own catalog organization, the source organization is set as the catalog organization of the new organization.

4. Click **Child Organizations** and **Create New** to add Child Organizations to the store (Figure 7-26).

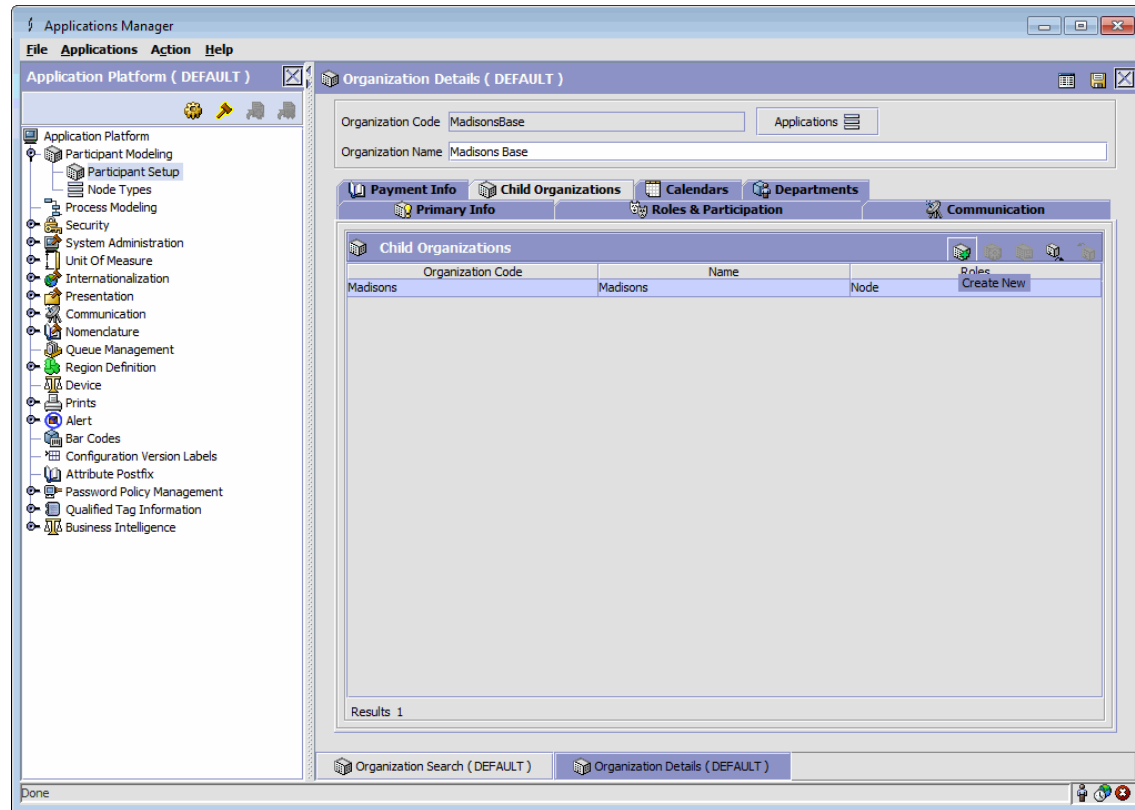


Figure 7-26 Application Manager Organization Details Create New Child Organization window

5. Add the following details in the Create Organization window (Figure 7-27):
- Organization Code: Madisons (primary fulfillment Center - online only)
 - Organization Name: Madisons
 - Primary Enterprise: MadisonsBase
 - Select the **Organization is a Fulfillment Node** check box.
 - From the Node type drop-down menu select **Store**.

Click **Save** for each fulfillment center.

The screenshot shows the 'Create Organization' window in the Applications Manager. The window has a title bar 'Applications Manager' and a subtitle 'Create Organization'. The main area contains the following fields and controls:

- Organization Code: Text box with 'Madisons'
- Organization Name: Text box with 'Madisons'
- DUNS Number: Empty text box
- Account Number With Hub: Empty text box
- Locale: Dropdown menu with 'US Eastern time' and a globe icon
- Parent Organization: Dropdown menu with 'MadisonsBase'
- ☐ Organization Is An Enterprise
- Primary Enterprise: Dropdown menu with 'MadisonsBase'
- ☒ Organization is a Fulfillment Node
- Node Type: Dropdown menu with 'Store' and a green plus icon
- ☐ Organization is a Yard

At the bottom, there are two tabs: 'Address' (selected) and 'Contact Info'. The 'Address' tab is currently empty.

Figure 7-27 Application Manager Create Organization Detail Window pane

7.5.2 Catalog management

The catalog management of SSFS aggregates and manages detailed product and catalog data across multiple divisions, enterprises, and participants. It acts as a multi-tenant management tool that supports sharing and collaboration. It enables categorization, product cross-sell, up-sell, substitution, and other features. To use catalog management, perform these steps:

1. Open the **MadisonsBase** store (Figure 7-28) that we created and click **Catalog Management** from the Applications menu.

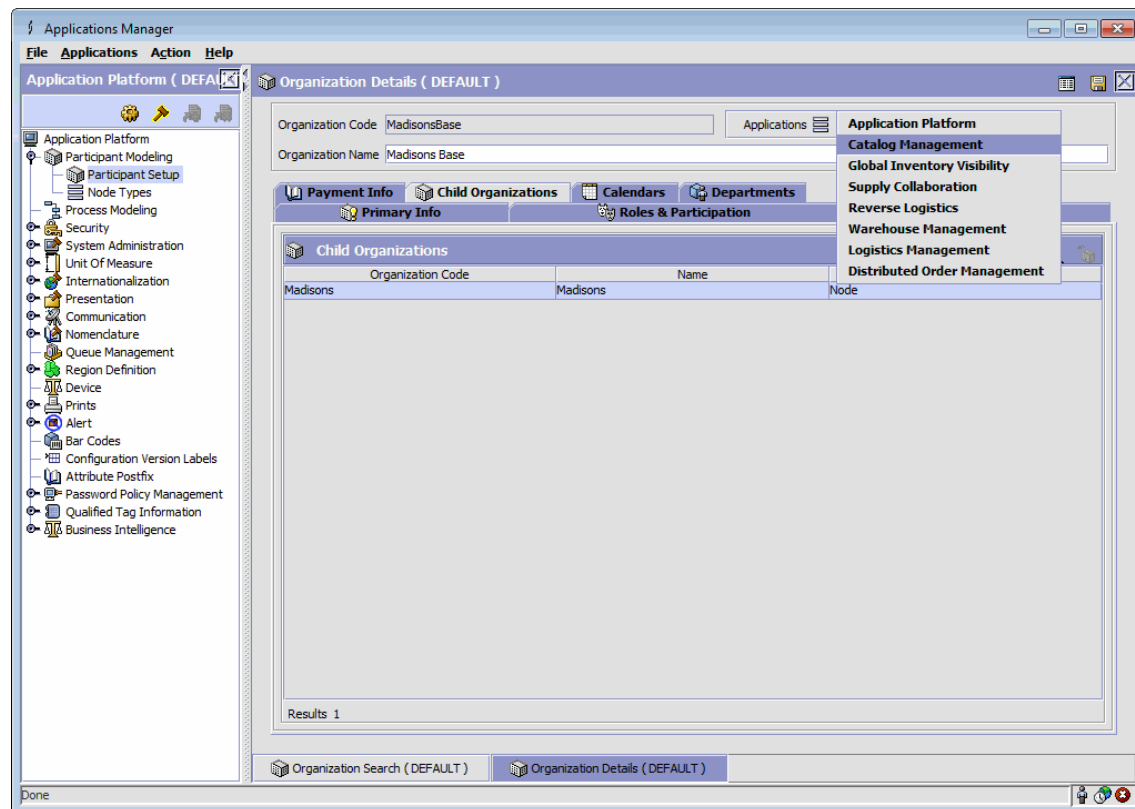


Figure 7-28 Application Manager Catalog Management

2. Open **Item UOM Master** (Figure 7-29) under Products and click **Create new**.

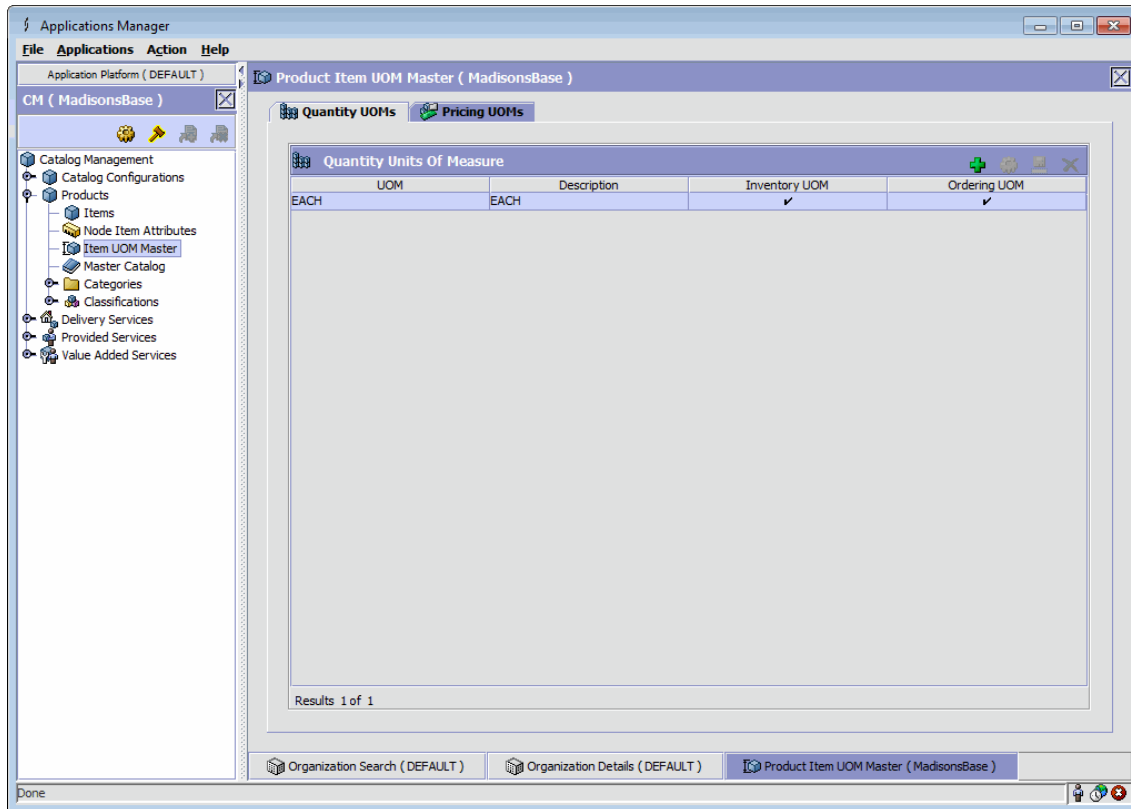


Figure 7-29 Application Manager UOM master

3. This opens the new UOM Details window. Enter the details shown in Figure 7-30 to ensure that the store has the unit of measurement (UOM) **EACH**. Select both the **Inventory Is Stored In This UOM** and **Orders Can Be Placed Using This UOM** check boxes, and then save.

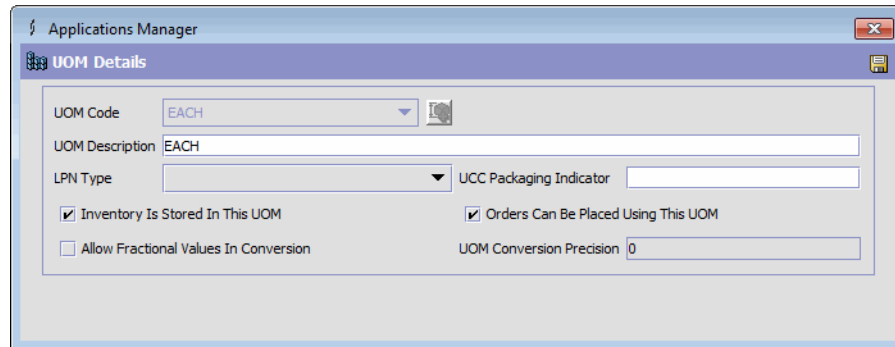


Figure 7-30 Application Manager UOM Details pane

7.5.3 Global Inventory Visibility application

The Sterling Global Inventory Visibility application is a collection of common components used to define inventory and capacity availability throughout the system.

In the Applications Manager you can use the Global Inventory Visibility configuration grouping to establish the following aspects of the Selling and Fulfillment Foundation for your business applications:

- ▶ Inventory rules
- ▶ Inventory types and considerations
- ▶ Distribution rules
- ▶ Resource capacity

Note: For more information about Global Inventory Visibility Configuration, see section 1.2 of the *Sterling Global Inventory Visibility: Configuration Guide*, which comes with the product.

To set the Sterling Global Inventory configuration:

1. Open the **MadisonsBase** store, click **Application**, then click **Global inventory Visibility** (Figure 7-31).

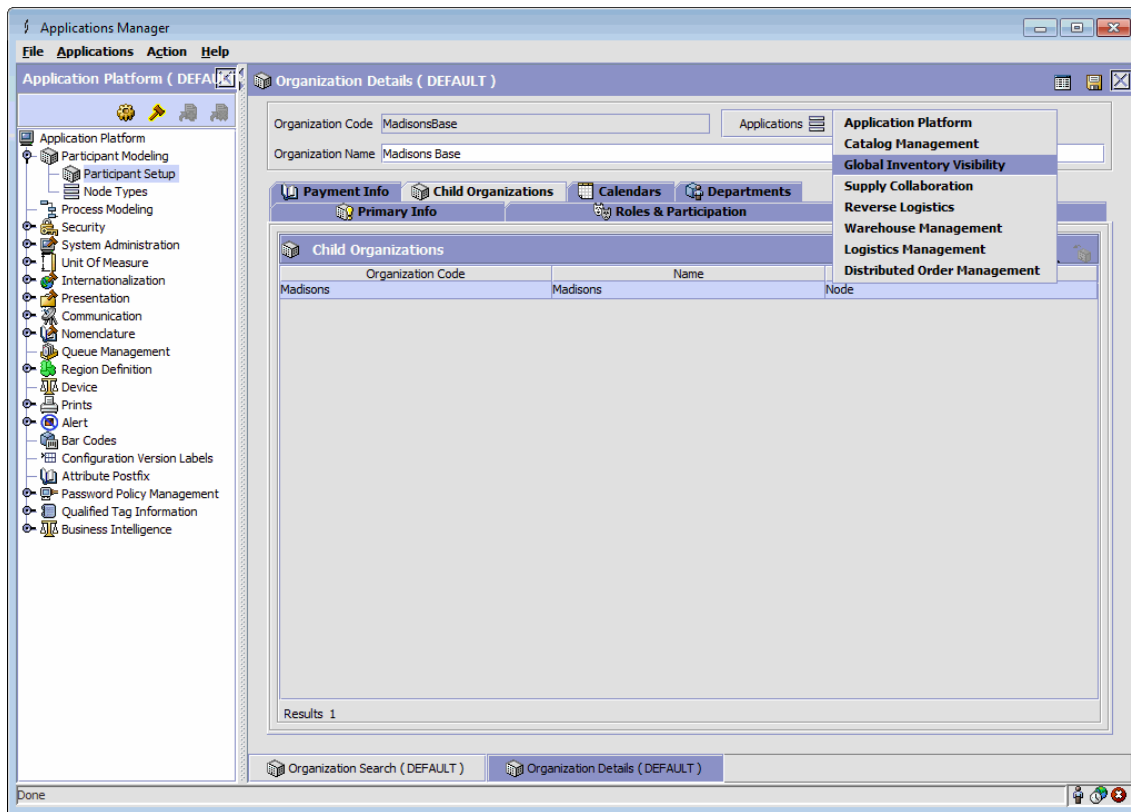


Figure 7-31 Application Manager Organization Details window

2. Open the Inventory Rules window (Figure 7-32), and click **Create New**.

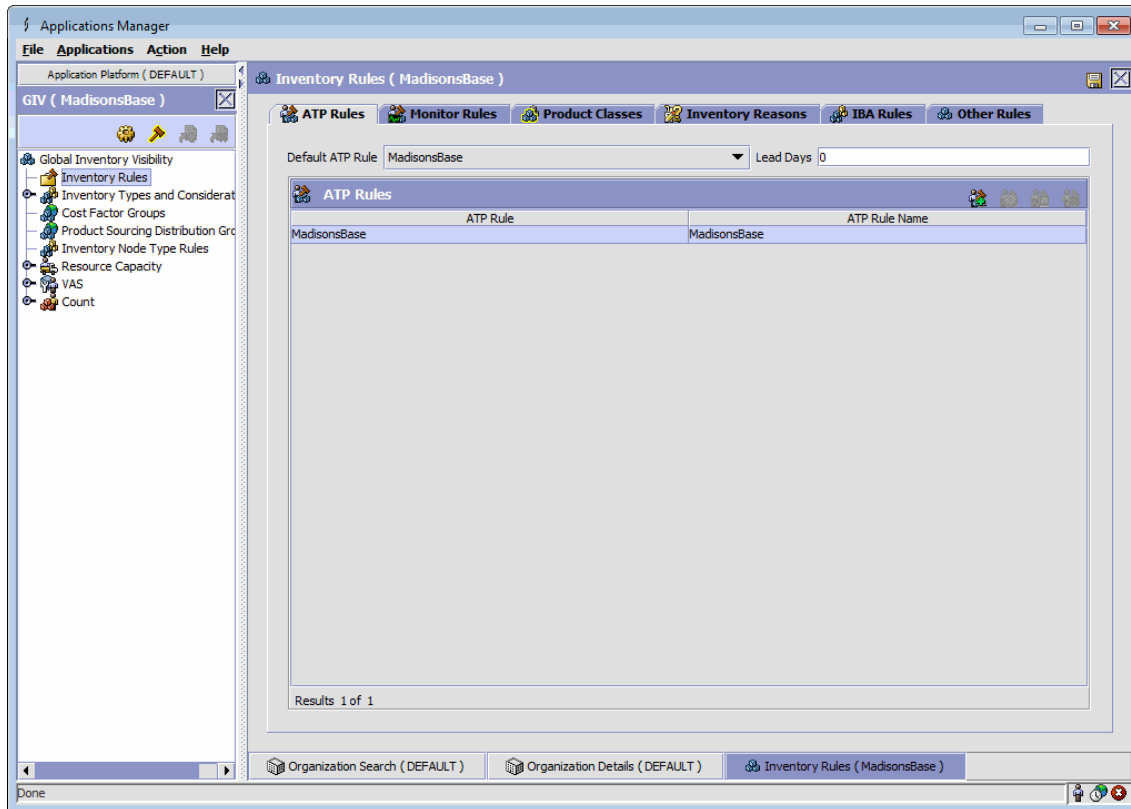


Figure 7-32 Application Manager Inventory Rules

3. Add ATP rule details (Figure 7-33). Ensure that the parameters are set to the following values:
 - ATP Rule: MadisonsBase
 - ATP Rule Name: MadisonsBase
 - ATP Rule Information: Set everything to 100 in this section.

Click **Save**.

Primary Information	
ATP Rule	MadisonsBase
ATP Rule Name	MadisonsBase

ATP Rule Information	
Past Due Supply Days	100
Past Due Demand Days	100
Forward Consumption (Days)	100
Backward Consumption (Days)	100
Processing Time For Future Inventory (Days)	100
Advance Notification Time (Days)	100
Accumulation Time (Days)	100

Figure 7-33 Application Manager ATP Rule Details pane

4. Set the default ATP Rule to the rule created (Figure 7-32 on page 171). Close the inner window.

Creating an item in the catalog

After creating the organization structure and catalog management, add items and set the other attributes:

1. Click **Application** → **Catalog Management** and click **Items** from the Products menu (Figure 7-34).

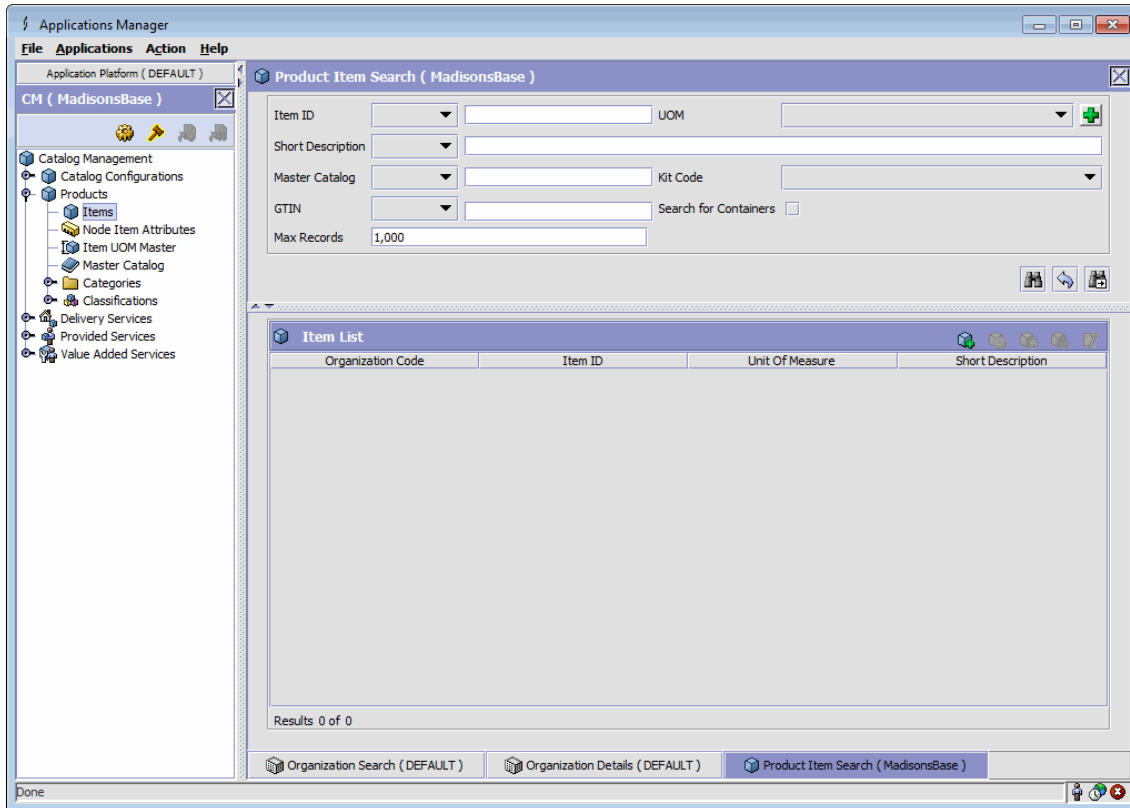


Figure 7-34 Application Manager Create Item

2. Click **Create new** to open the Create Item window (Figure 7-35).

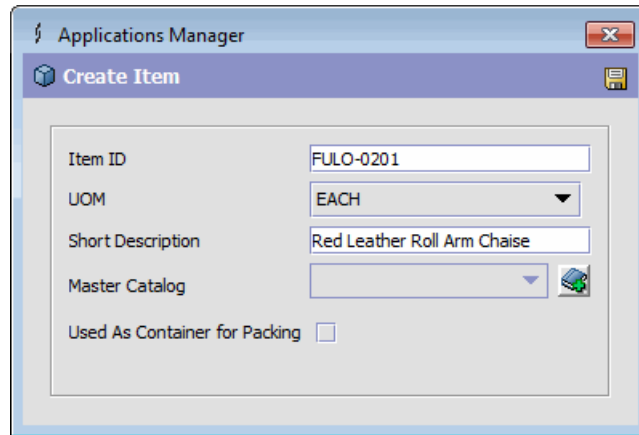


Figure 7-35 Application Manager Create Item details

3. Enter the item details for the item that you are creating in SSFS (Figure 7-35). While creating an item, ensure that the following fields are populated:
 - Item ID: This must match the WebSphere Commerce Item SKU (partnumber).
 - UOM: EACH
 - Short Description: This should match the commerce item.
 - Click **Save** and change the status to **Published**.

Click **Save** and **Close**.

Important: This would usually be a data load operation from a catalog master. Repeat the same steps to create multiple items.

Note: Creating an item in the catalog section can also be performed via the Sterling Selling and Fulfillment Suite: Business Console at:

<http://<host>/sbc/sbc/login.do>

7.5.4 Distributing Order Management

The Selling and Fulfillment Foundation Distributed Order Management application provides highly configurable order management capabilities for all types of customer orders (products and services). It aggregates, manages, and

monitors orders from all channels, and coordinates fulfillment processes across the extended enterprise. Distributed Order Management (DOM) checks for inventory availability and provides rule-based, dynamic allocation across all internal and external fulfillment locations. Moreover, it coordinates critical third-party services, such as credit, logistics, and installation, and enables collaborative execution among all involved participants. It provides a single order repository and allows customers, channels, suppliers, and trading partners access to real-time order information throughout the entire fulfillment life cycle. DOM delivers complete flexibility for handling multiple order fulfillment processes in a single instance and handles dynamic variations in order processes with event-driven and rule-based order coordination.

To set the financial attributes:

1. Click **Distribute Order Management** from the Applications menu (Figure 7-36).

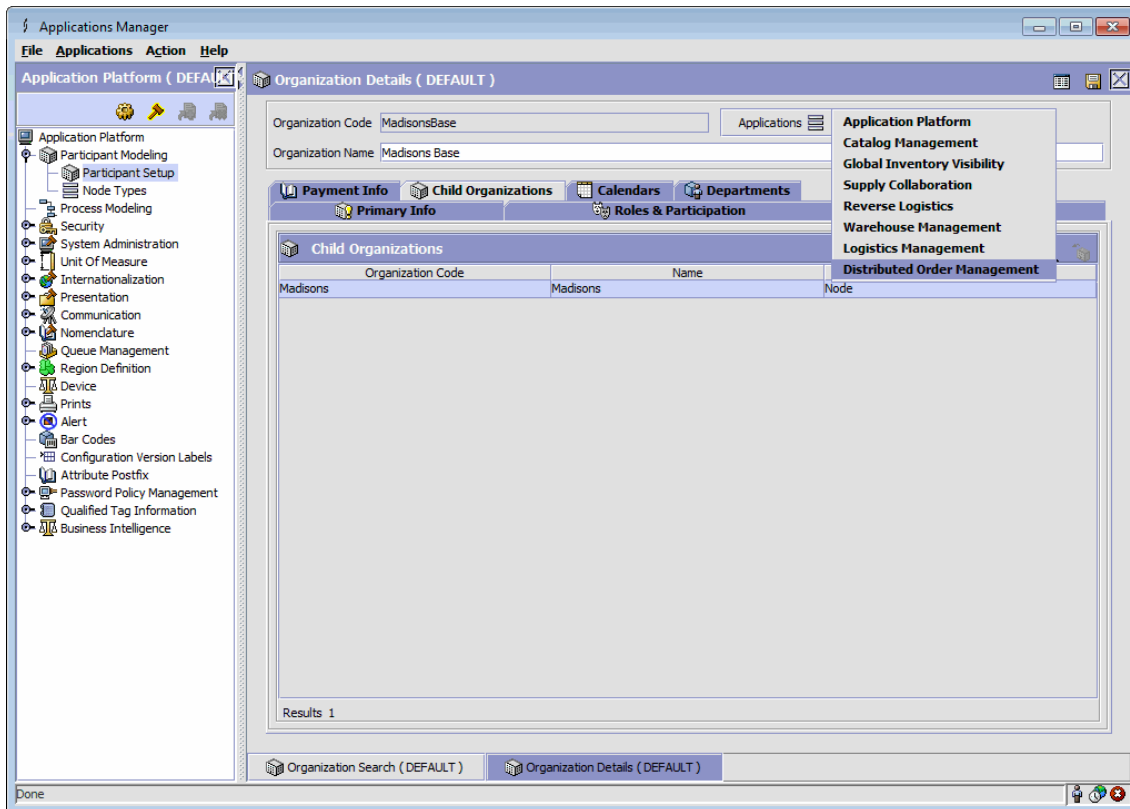


Figure 7-36 Application Manager Organization Details

2. This action opens the DOM section on the left side of the window (Figure 7-37). Click **Document Specific** → **Sales Order** → **Financials** → **Financial Attributes** to open the Financial Attributes window.

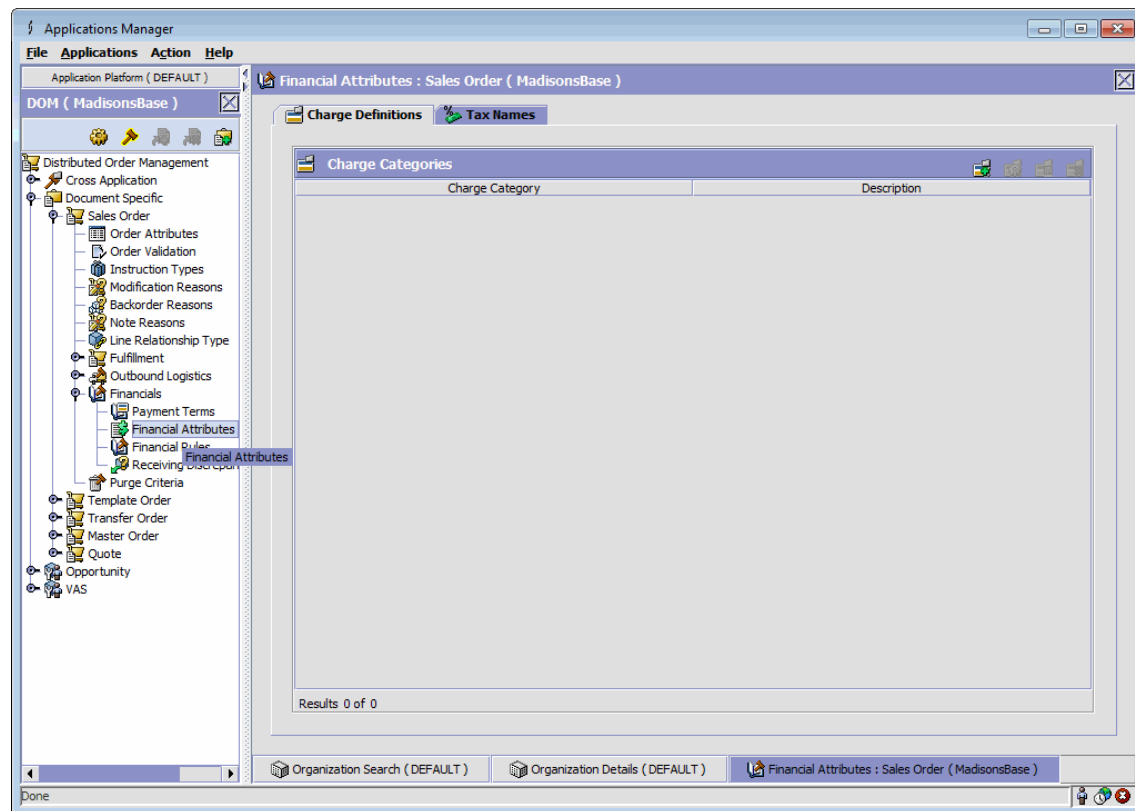


Figure 7-37 Application Manager Financial Attributes

3. Click **Create New** on the Charge Definitions tab. Enter the details in the Charge Categories and Description fields (Figure 7-38) and select the **Billable** check box.

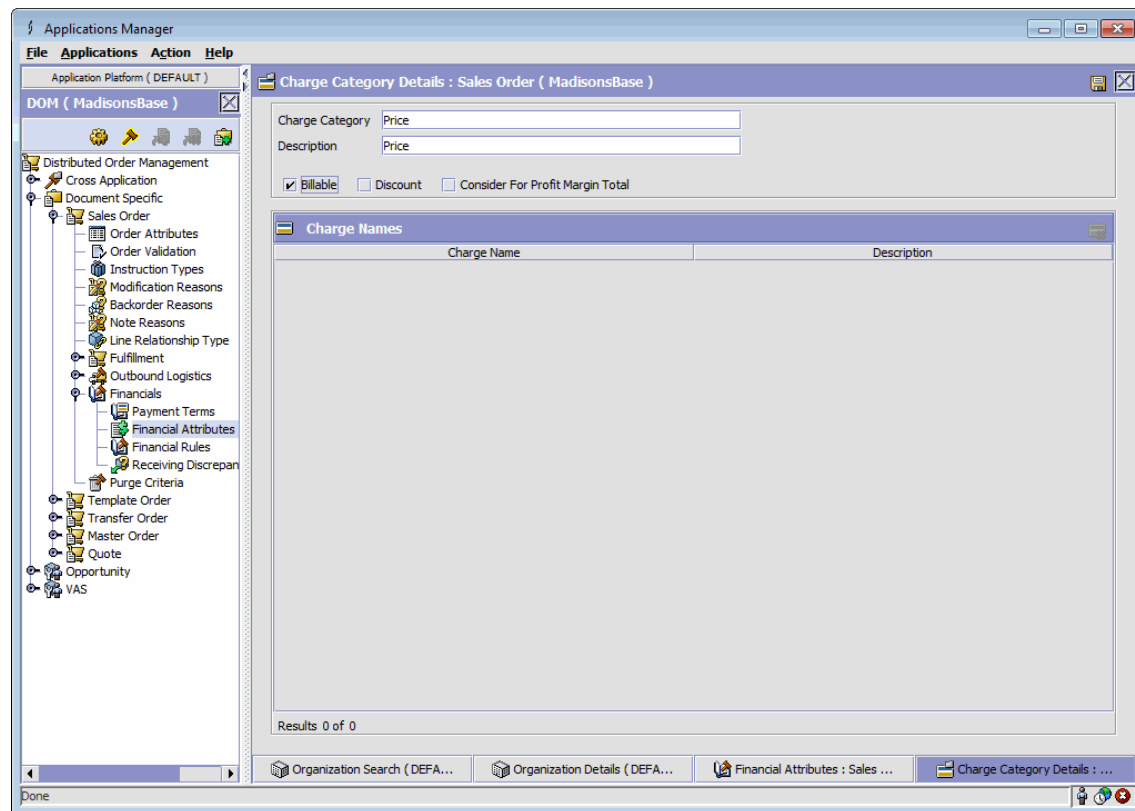


Figure 7-38 Application Manager Charge Categories

- Click **Create new**. On the Charge Name Details window (Figure 7-39) add the charge name details. Note that the Name field can be blank.

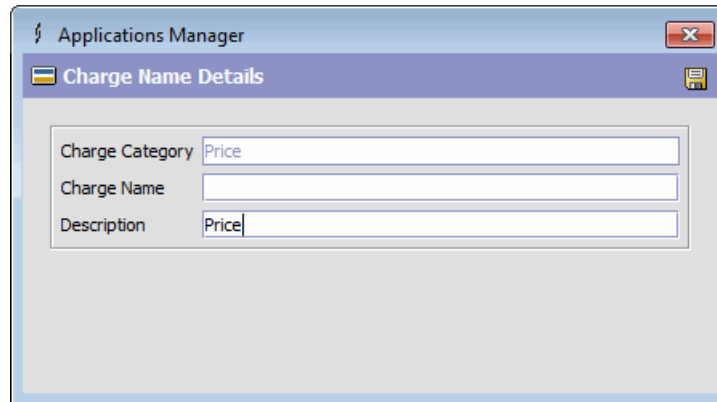


Figure 7-39 Application Manager Charge Name Details

- Close the page and add two more similar charge categories, **Discount** and **Shipping** (using steps 3 and 4). Ensure that while creating charge details for the discount, both **Discount** and **Billable** check boxes are selected. For shipping, only the **Billable** check box should be selected. Add a charge name of discount under Charge Categories Discount and add charge names as Shipping and Shipping Charge.
- In Figure 7-37 on page 176, click the **Tax Names** tab. For the Tax Name Details fields, enter Sales Tax (Figure 7-40). Save and then close the window.

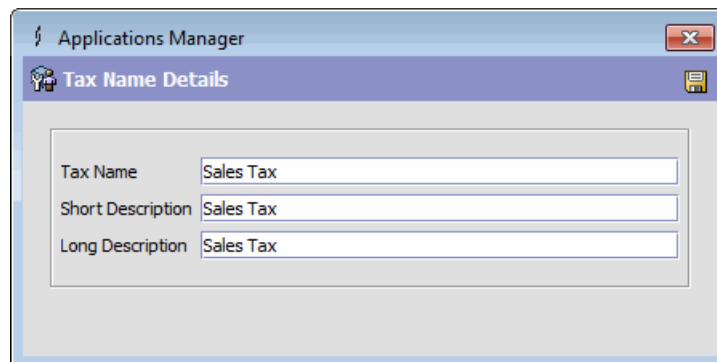


Figure 7-40 Application Manager Tax name Details

Ensure that you are still in DOM (Figure 7-37 on page 176).

- Open the Payment Types window (Figure 7-41) and select **Cross Application** → **Financials** → **Payment Types** on the left, and then click

Create New. The payment type will be entered in the page shown in Figure 7-42 on page 180.

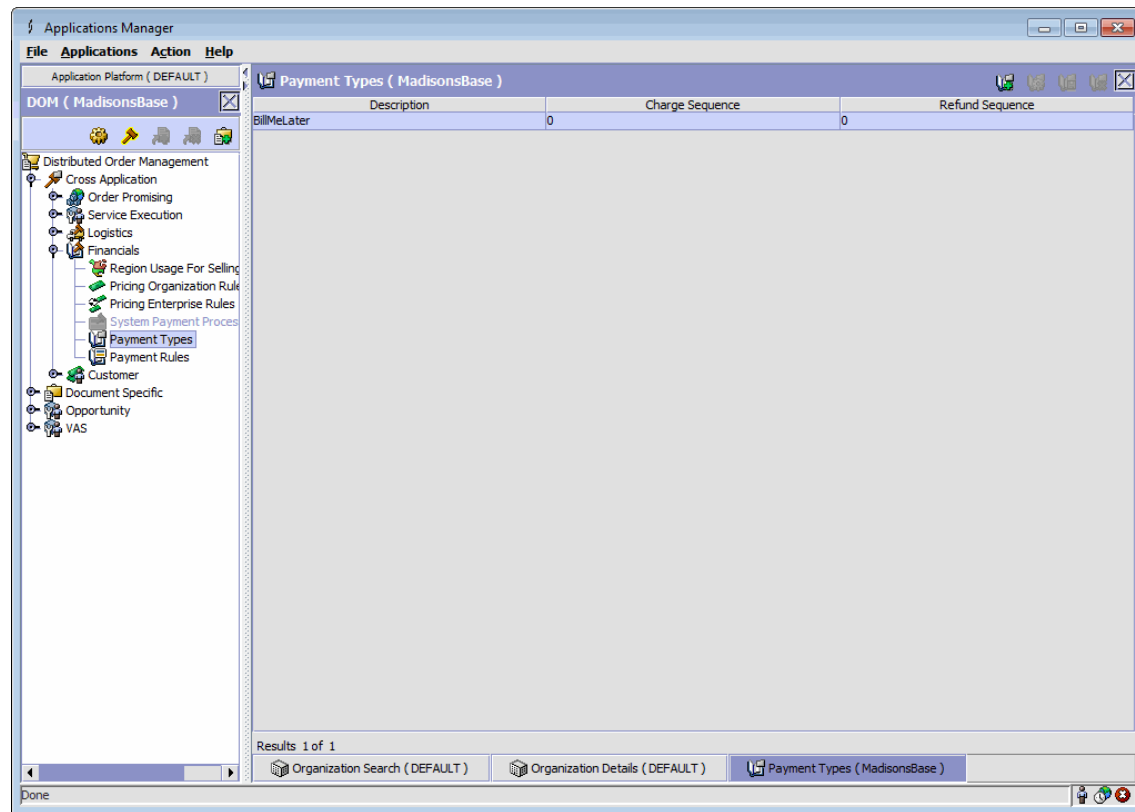


Figure 7-41 Application Manager Payment Types window

8. Create the payment types in SSFS (Figure 7-42).

The screenshot shows a window titled "Applications Manager" with a sub-header "Payment Type Details". Inside the window, there are two tabs: "Charge" (selected) and "Refund". Under the "Charge" tab, there are three input fields: "Payment Type" (containing "COD"), "Payment Type Group" (a dropdown menu showing "Other"), and "Description" (containing "COD"). Below these fields, there is a section for "Charge Sequence" with a text box. Underneath, there are three checkboxes: "Charge Instead of Authorize", "Charge Up To Available", and "Charge Consolidation Allowed". At the bottom, there is a "Consolidation Window (Hrs)" label followed by a text box.

Figure 7-42 Application Manager Payment Type Details

Repeat this step for all the other payment types in WebSphere Commerce.

Important: All the payment types must match the paymentMethod parameter in the WebSphere Commerce PaymentMappings.xml file.

9. After you are finished with all the payment types, close the Distributed Order Management option.

Adding inventory to an item

New items are created in SSFS with tax and payment details. Follow the steps below to add inventory. Remember that inventory needs to be added for each SKU, for each fulfillment Center.

1. Start Sterling Selling and Fulfillment Suite: Application Console from:
`http://<hostname>/smcfs/console/login.jsp`
2. Click the **Inventory** tab and select the **Adjust Inventory** menu (Figure 7-43).

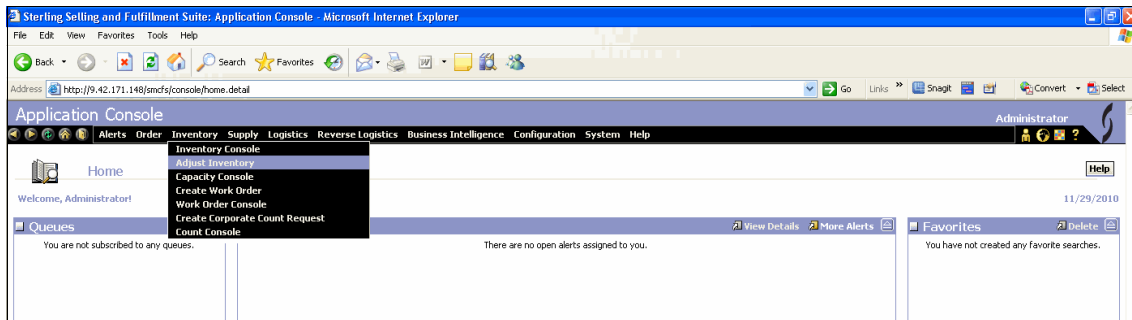


Figure 7-43 Application Console window

3. Select the proper organization from the available drop-down option and provide the item ID that needs to be adjusted (Figure 7-43) to add/edit inventory in the item.
4. Click **Proceed**.

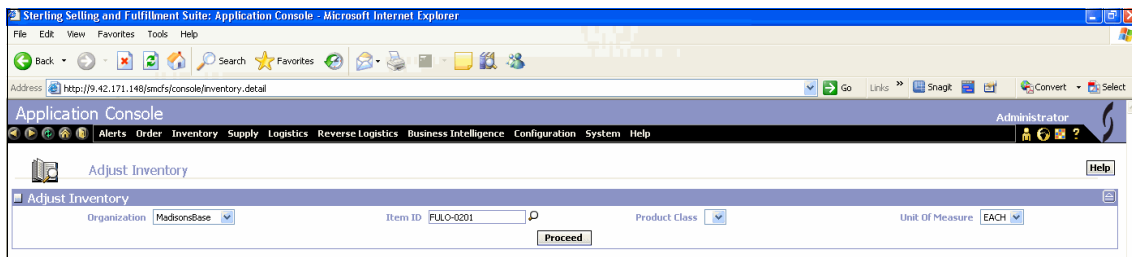


Figure 7-44 Application Console Adjust Inventory window

5. Go to Adjustments section and in the Quantity drop-down select the **Increased By** option. Enter the inventory value to add (Figure 7-45) and click **Save**.

The screenshot shows the 'Adjust Inventory' page in the Sterling Selling and Fulfillment Suite Application Console. The page is titled 'Adjust Inventory' and has a 'Save' button. It is divided into several sections:

- Item Details:** Item ID: FULO-0201, Product Class, Unit of Measure: EACH.
- Inventory Information:** Ship Node, Supply Type, Segment Type, Segment.
- Adjustments:** Availability: Track (selected), Quantity: 0.00, Increase By: 100.
- References:** Reference #1, Reference #2, Reference #3, Reference #4, Reference #5.
- Modification Reason:** Reason Code, Reason Text.

Figure 7-45 Application Console Inventory Adjustments

After admin configuration is done, SSFS is ready for integration with WebSphere Commerce for the available business scenarios.

7.6 Integration flow data mapping

The main flow of data between WebSphere Commerce and SSFS is the same as with any other DOM system. Data flows through the two systems. The integration flow is outlined in the DOM documentation. For details see “Distributed Order Management (DOM) integration flows” at the WebSphere Commerce Infocenter:

<http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.dom-integration.doc/concepts/csmdomintegrationflow.htm>

Figure 7-46 shows the flow.

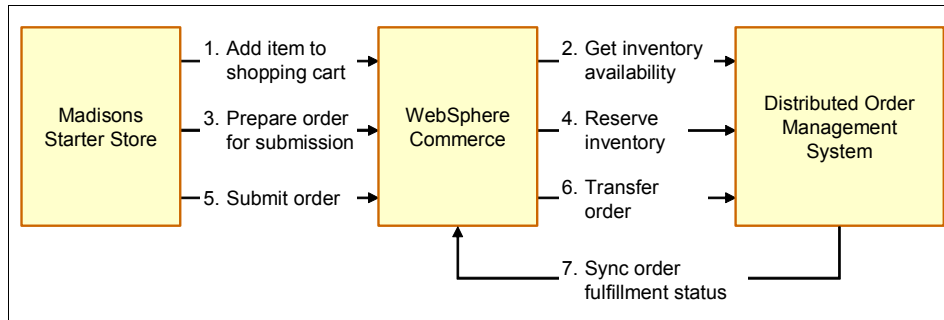


Figure 7-46 Integration flow image

The supported integration flows are implemented, taking into account several assumptions, such as:

- ▶ One ship node per store (physical or online).
- ▶ WebSphere Commerce handles payments
- ▶ The catalog does not contain bundles or kits.

These assumptions can all be enabled with additional customizing of the solution.

General mapping rules

Each installation of WebSphere Commerce and SSFS will be different. Some assumptions were made about the mapping between the two systems, and these assumptions are reflected in the WESB mediation module. Of course, the module will have to be changed to suit your particular environment. Table 7-12 lists the mapping assumptions.

Table 7-12 General mapping rules

WebSphere Commerce field	SSFS field
UOM - C62	UOM - EACH
Item Part Number	Item Id
Online and Physical Store Names	Shipnode Identifier
Shipping Mode Code	Carrier Service Code
Fulfillment Center name	Shipnode identifier
Order Id	Inventory Reservation Id = "WC_" + WC Order No

WebSphere Commerce field	SSFS field
Order Id	SSFS Order No = "WC_" + WC Order Id
Order Item Id	Order Line No

Note: These rules are a general overview and reflect the current implementation only. The exact mappings are listed in the following sections in detail and are subject to change.

Get inventory availability communication

The *get inventory availability* communication maps the WebSphere Commerce GetInventoryAvailability request to the SSFS findInventory API call. It is possible to cache the result. This approach drastically increases performance, as a real-time call to SSFS is not required for each inventory check. Inventory caching is handled only in a pull that is initiated by WebSphere Commerce. A more robust method would be for SSFS to push inventory changes.

As both WebSphere Commerce and SSFS are very customizable, assumptions had to be made with regards to the mapping from one system to the other.

Table 7-13 identifies the request information for this call.

Table 7-13 Request

SSFS API path	Source
Promise/@OrganizationCode	Set in the "OrganizationCodeSetter" node in the mediation flow
Promise/PromiseLines/PromiseLine@UnitOfMeasure	Always EACH
Promise/PromiseLines/PromiseLine@RequiredQty	Always 999999999
Promise/PromiseLines/PromiseLine@ItemID	PartNumber param in the Xpath selection criteria
Promise/PromiseLines/PromiseLine@ShipNode	<ul style="list-style-type: none"> ▶ NamelIdentifier in the Xpath selection criteria (if present) or ▶ ExternalIdentifier in the Xpath selection criteria the online or physical store names
Promise/PromiseLines/PromiseLine@LineId	<ul style="list-style-type: none"> ▶ online_# or physical_# (Where # is a number appended to ensure uniqueness.)

Table 7-14 lists the response (results).

Table 7-14 Response

WC path	Source
_inv:ShowInventoryAvailability/_inv:DataArea/_inv:InventoryAvailability/_inv:InventoryAvailabilityIdentifier/_inv:ExternalIdentifier/_inv:CatalogEntryIdentifier/_wcf:ExternalIdentifier/_wcf:PartNumber	PartNumber param in the Xpath selection criteria in the request XML
_inv:ShowInventoryAvailability/_inv:DataArea/_inv:InventoryAvailability/_inv:InventoryAvailabilityIdentifier/_inv:ExternalIdentifier/_inv:OnlineStoreIdentifier/_wcf:ExternalIdentifier/_wcf:NameIdentifier	<ul style="list-style-type: none"> ▶ NameIdentifier in the XPath selection criteria in the request XML (if present) or <ul style="list-style-type: none"> ▶ ExternalIdentifier in the XPath selection criteria in the request XML
_inv:ShowInventoryAvailability/_inv:DataArea/_inv:InventoryAvailability/_inv:InventoryStatus	<ul style="list-style-type: none"> ▶ Available: If available now according to product availability date ▶ Backorderable: If available in the future according to product availability date ▶ Unavailable: All other cases
_inv:ShowInventoryAvailability/_inv:DataArea/_inv:InventoryAvailability/_inv:AvailableQuantity@uom	Always "C62"
_inv:ShowInventoryAvailability/_inv:DataArea/_inv:InventoryAvailability/_inv:AvailableQuantity	Assignments/Assignment/@Quantity (A summation of all quantity values for available products)

Reserve inventory communication

The *reserve inventory* communication maps the WebSphere Commerce ProcessInventoryRequirement request to the SSFS reserveAvailableInventory API call. As both WebSphere Commerce and SSFS are very customizable, assumptions had to be made with regard to the mapping from one system to the other. Table 7-15 shows the reserve inventory request information.

Table 7-15 Request

SSFS API path	Source context	Source
Promise/@OrganizationCode	N/A	Set in the "OrganizationCodeSetter" node in the mediation flow
Promise/ReservationParameters/@ReservationID	inv:DataArea/_inv:InventoryRequirement	"WC_" prepended to _ord:OrderIdentifier/_wcf:Unique ID

SSFS API path	Source context	Source
Promise/PromiseLine@LineID	inv:DataArea/_inv:InventoryRequirement/_ord:OrderItem	"WC_" prepended to _ord:OrderItemIdentifier/_wcf:UniqueID
Promise/PromiseLine@ItemID	inv:DataArea/_inv:InventoryRequirement/_ord:OrderItem	_ord:CatalogEntryIdentifier/_wcf:ExternalIdentifier/_wcf:PartNumber
Promise/PromiseLine@RequiredQty	inv:DataArea/_inv:InventoryRequirement/_ord:OrderItem	_ord:Quantity
Promise/PromiseLine@FillQuantity	inv:DataArea/_inv:InventoryRequirement/_ord:OrderItem	_ord:Quantity
Promise/PromiseLine@UnitOfMeasure	inv:DataArea/_inv:InventoryRequirement/_ord:OrderItem	_ord:Quantity/@uom If above is "C62" than EACH, otherwise, use the value of above
Promise/PromiseLine@CarrierServiceCode	inv:DataArea/_inv:InventoryRequirement/_ord:OrderItem	_ord:OrderItemShippingInfo/_ord:ShippingMode/_ord:ShippingModelIdentifier/_ord:ExternalIdentifier/_ord:ShipModeCode If above is empty than "Priority", otherwise, use the value of above
Promise/PromiseLine@ShipNode	inv:DataArea/_inv:InventoryRequirement/_ord:OrderItem	_ord:FulfillmentCenter/_ord:FulfillmentCenterIdentifier/_wcf:Name
Promise/PromiseLine/ShipToAddress@AddressLine1	inv:DataArea/_inv:InventoryRequirement/_ord:OrderItem	_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:AddressLine[1]
Promise/PromiseLine/ShipToAddress@AddressLine2	inv:DataArea/_inv:InventoryRequirement/_ord:OrderItem	_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:AddressLine[2]
Promise/PromiseLine/ShipToAddress@AddressLine3	inv:DataArea/_inv:InventoryRequirement/_ord:OrderItem	_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:AddressLine[3]
Promise/PromiseLine/ShipToAddress@City	inv:DataArea/_inv:InventoryRequirement/_ord:OrderItem	_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:City
Promise/PromiseLine/ShipToAddress@State	inv:DataArea/_inv:InventoryRequirement/_ord:OrderItem	_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:StateOrProvinceName

SSFS API path	Source context	Source
Promise/PromiseLine/ShipToAddress@Country	inv:DataArea/_inv:InventoryRequirement/_ord:OrderItem	_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:Country
Promise/PromiseLine/ShipToAddress@ZipCode	inv:DataArea/_inv:InventoryRequirement/_ord:OrderItem	_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:PostalCode

Table 7-16 shows the response information.

Table 7-16 Response

WC Path	Source Context	Source
_inv:AcknowledgeInventoryRequirement/_inv:DataArea/_inv:InventoryRequirement/_ord:OrderIdentifier/_wcf:UniqueID	N/A	PromiseLines/PromiseLine/Reservations/Reservation/@ReservationID With the "WC_" removed from the start if it exists
_inv:AcknowledgeInventoryRequirement/_inv:DataArea/_inv:InventoryRequirement/_ord:OrderItem/_ord:OrderItemIdentifier/_wcf:UniqueID	PromiseLines/PromiseLine	@ItemID
_inv:AcknowledgeInventoryRequirement/_inv:DataArea/_inv:InventoryRequirement/_ord:OrderItem/_ord:OrderItemStatus/_ord:InventoryStatus	PromiseLines/PromiseLine	Reservations/Reservation/@ReservationID "Allocated" if availability date before today, "Backordered" if availability date after today, otherwise "Unallocated"
_inv:AcknowledgeInventoryRequirement/_inv:DataArea/_inv:InventoryRequirement/_ord:OrderItem/_ord:OrderItemFulfillmentInfo/_ord:AvailableDate (Optional)	PromiseLines/PromiseLine	Reservations/Reservation/@ProductAvailabilityDate Only set if the ProductAvailabilityDate attribute is set in the message
_inv:AcknowledgeInventoryRequirement/_inv:DataArea/_inv:InventoryRequirement/_ord:OrderItem/_ord:OrderItemFulfillmentInfo/_ord:ExpectedShipDate (Optional)	PromiseLines/PromiseLine	Reservations/Reservation/@ShipDate Only set if the ShipDate attribute is set in the message.

WC Path	Source Context	Source
_inv:AcknowledgeInventoryRequirement/_inv:DataArea/_inv:InventoryRequirement/_ord:OrderItem/_ord:FulfillmentCenter/_ord:FulfillmentCenterIdentifier/_wcf:UniqueID (Optional)	PromiseLines/PromiseLine	@ShipNode Only if present
_inv:AcknowledgeInventoryRequirement/_inv:DataArea/_inv:InventoryRequirement/_ord:OrderItem/_ord:FulfillmentCenter/_ord:FulfillmentCenterIdentifier/_wcf:Name (Optional)	PromiseLines/PromiseLine	@ShipNode Only if present

Transfer order communication

The *transfer order* communication maps the WebSphere Commerce ProcessOrder request to the SSFS createOrder API call. As both WebSphere Commerce and SSFS are very customizable, assumptions had to be made with regard to the mapping from one system to the other. Table 7-17 presents the transfer order request for this call.

Table 7-17 Request

SSFS API path	Source context	Source
Order/@EnterpriseCode	N/A	Set in the “OrganizationCodeSetter” node in the mediation flow
Order/@SellerOrganizationCode	N/A	Set in the “OrganizationCodeSetter” node in the mediation flow
Order/@OrderDate	_ord:DataArea/_ord:Order	_ord:PlacedDate
Order/OrderLines/OrderLine/@PrimeLineNo	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:OrderItemIdentifier/_wcf:UniqueID If above contains a “-”, use only part before the dash
Order/OrderLines/OrderLine/@SubLineNo (Not always present)	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:OrderItemIdentifier/_wcf:UniqueID If above contains a “-”, use only part after the dash, otherwise, this field is blank

Order/OrderLines/OrderLine/@OrderedQty	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:Quantity
Order/OrderLines/OrderLine/@CarrierServiceCode	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:OrderItemShippingInfo/_ord:ShippingMode/_ord:ShippingModelIdentifier/_ord:ExternalIdentifier/_ord:ShipModeCode
Order/OrderLines/OrderLine/@ShipNode	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:FulfillmentCenter/_ord:FulfillmentCenterIdentifier/_wcf:Name
Order/OrderLines/OrderLine/@ReqShipDate	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:OrderItemShippingInfo/_ord:RequestedShipDate
Order/OrderLines/OrderLine/@DeliveryMethod	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:OrderItemShippingInfo/_ord:ShippingMode/_ord:ShippingModelIdentifier/_ord:ExternalIdentifier/_ord:ShipModeCode If above is "PickupInStore" then "PICK", otherwise "SHP"
Order/OrderLines/OrderLine@FillQuantity	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:Quantity
Order/OrderLines/OrderLine/OrderLineReservations/OrderLineReservation/@ItemID	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:CatalogEntryIdentifier/_wcf:ExternalIdentifier/_wcf:PartNumber
Order/OrderLines/OrderLine/OrderLineReservations/OrderLineReservation/@Node	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:FulfillmentCenter/_ord:FulfillmentCenterIdentifier/_wcf:Name
Order/OrderLines/OrderLine/OrderLineReservations/OrderLineReservation/@Quantity	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:Quantity
Order/OrderLines/OrderLine/OrderLineReservations/OrderLineReservation/@ReservationID	_ord:DataArea/_ord:Order/_	"WC_" prepended to _ord:OrderIdentifier/_wcf:UniqueID
Order/OrderLines/OrderLine/Item/@ItemID	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:CatalogEntryIdentifier/_wcf:ExternalIdentifier/_wcf:PartNumber
Order/OrderLines/OrderLine/Item/@UnitOfMeasure	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:Quantity/@uom If above is "C62" than "EACH", otherwise, use the value of above

Order/OrderLines/OrderLine/PersonInfoShipTo@FirstName	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:ContactName/_wcf::FirstName
Order/OrderLines/OrderLine/PersonInfoShipTo@LastName	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:ContactName/_wcf::LastName
Order/OrderLines/OrderLine/PersonInfoShipTo@AddressLine1	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:Addresses/_wcf::AddressLine[1]
Order/OrderLines/OrderLine/PersonInfoShipTo@AddressLine2	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:Addresses/_wcf::AddressLine[2]
Order/OrderLines/OrderLine/PersonInfoShipTo@AddressLine3	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:Addresses/_wcf::AddressLine[3]
Order/OrderLines/OrderLine/PersonInfoShipTo@City	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:Addresses/_wcf::City
Order/OrderLines/OrderLine/PersonInfoShipTo@State	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:Addresses/_wcf::StateOrProvinceName
Order/OrderLines/OrderLine/PersonInfoShipTo@Country	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:Addresses/_wcf::Country
Order/OrderLines/OrderLine/PersonInfoShipTo@ZipCode	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:Addresses/_wcf::ZipCode
Order/OrderLines/OrderLine/PersonInfoShipTo@EmailID	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:EmailAddress1
Order/OrderLines/OrderLine/LinePriceInfo@UnitPrice	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:OrderItemAmount/_wcf:UnitPrice/_wcf:Price
Order/OrderLines/OrderLine/LinePriceInfo@ListPrice	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:OrderItemAmount/_wcf:UnitPrice/_wcf:Price
Order/OrderLines/OrderLine/LinePriceInfo@RetailPrice	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:OrderItemAmount/_wcf:UnitPrice/_wcf:Price

Order/OrderLines/OrderLine/LinePriceInfo@IsLinePriceForInformationOnly		N
Order/OrderLines/OrderLine/LinePriceInfo@PriceQuantityStrategy		FIX
Order/OrderLines/OrderLine/LinePriceInfo@IsPriceLocked		Y
Order/OrderLines/OrderLine/LineCharges/LineCharge[1]/@ChargePerLine	_ord:DataArea/_ord:Order/_ord:OrderItem/_ord:OrderItemAmount/_wcf:Adjustment	_wcf::Amount
Order/OrderLines/OrderLine/LineCharges/LineCharge[1]/@ChargeCategory		DISCOUNT
Order/OrderLines/OrderLine/LineCharges/LineCharge[2]/@ChargePerLine	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:OrderItemAmount/_wcf:ShippingCharge
Order/OrderLines/OrderLine/LineCharges/LineCharge[2]/@ChargeCategory		Shipping
Order/OrderLines/OrderLine/LineCharges/LineCharge[2]/@ChargeName		Shipping Charge
Order/OrderLines/OrderLine/LineTaxes/LineTax[1]/@ChargeCategory		Price
Order/OrderLines/OrderLine/LineTaxes/LineTax[1]/@Tax	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:OrderItemAmount/_wcf:SalesTax
Order/OrderLines/OrderLine/LineTaxes/LineTax[1]/@TaxName		Sales Tax
Order/OrderLines/OrderLine/LineTaxes/LineTax[2]/@ChargeCategory		Shipping
Order/OrderLines/OrderLine/LineTaxes/LineTax[2]/@Tax	_ord:DataArea/_ord:Order/_ord:OrderItem	_ord:OrderItemAmount/_wcf:ShippingTax
Order/OrderLines/OrderLine/LineTaxes/LineTax[2]/@TaxName		Shipping Tax

Order/PersonInfoBillTo@FirstName	_ord:DataArea/_ord:Order	_ord:OrderPaymentInfo/_ord:PaymentInstruction/_ord:BillingAddress/_wcf:ContactName/_wcf:FirstName
Order/PersonInfoBillTo@LastName	_ord:DataArea/_ord:Order	_ord:OrderPaymentInfo/_ord:PaymentInstruction/_ord:BillingAddress/_wcf:ContactName/_wcf:LastName
Order/PersonInfoBillTo@AddressLine1	_ord:DataArea/_ord:Order	_ord:OrderPaymentInfo/_ord:PaymentInstruction/_ord:BillingAddress/_wcf:Address/_wcf:AddressLine[1]
Order/PersonInfoBillTo@AddressLine2	_ord:DataArea/_ord:Order	_ord:OrderPaymentInfo/_ord:PaymentInstruction/_ord:BillingAddress/_wcf:Address/_wcf:AddressLine[2]
Order/PersonInfoBillTo@AddressLine3	_ord:DataArea/_ord:Order	_ord:OrderPaymentInfo/_ord:PaymentInstruction/_ord:BillingAddress/_wcf:Address/_wcf:AddressLine[3]
Order/PersonInfoBillTo@City	_ord:DataArea/_ord:Order	_ord:OrderPaymentInfo/_ord:PaymentInstruction/_ord:BillingAddress/_wcf:Address/_wcf:City
Order/PersonInfoBillTo@State	_ord:DataArea/_ord:Order	_ord:OrderPaymentInfo/_ord:PaymentInstruction/_ord:BillingAddress/_wcf:Address/_wcf:StateOrProvinceName
Order/PersonInfoBillTo@Country	_ord:DataArea/_ord:Order	_ord:OrderPaymentInfo/_ord:PaymentInstruction/_ord:BillingAddress/_wcf:Address/_wcf:Country
Order/PersonInfoBillTo@ZipCode	_ord:DataArea/_ord:Order	_ord:OrderPaymentInfo/_ord:PaymentInstruction/_ord:BillingAddress/_wcf:Address/_wcf:PostalCode
Order/PersonInfoBillTo@EmailID	_ord:DataArea/_ord:Order	_ord:OrderPaymentInfo/_ord:PaymentInstruction/_ord:BillingAddress/_wcf:EmailAddress1/

Table 7-18 shows the response for the transfer order.

Table 7-18 Response

WC Path	Source Context	Source
_ord:AcknowledgeOrder/_ord:DataArea/_ord:Order/_ord:OrderIdentifier/_wcf:UniqueID	N/A	@OrderNo

Sync order fulfillment status communication map

The *sync order fulfillment status* communication map is different from the previous methods, as it is a push from SSFS to WebSphere Commerce. It maps the SSFS getOrderDetails and SSFS getShipmentDetails APIs to the WebSphere Commerce SyncOrder message. As both WebSphere Commerce and SSFS are very customizable, assumptions had to be made with regard to the mapping from one system to the other. Table 7-19 shows the sync order fulfillment status request assumptions.

Table 7-19 getOrderDetailsRequest

WC path	Source	
_ord:OrderIdentifier/_wcf:UniqueID	@OrderNo The “WC_” is removed from the beginning if needed.	
_ord:OrderStatus/_ord:Status	@MaxOrderStatus The following table will choose the status:	
	WC status	SSFS order status
	R	>= 3200 and < 3700
	B	1300 or 1400

Table 7-20 lists the getShipmentDetails information.

Table 7-20 getShipmentDetails

WC path	Source
_ord:OrderIdentifier/_wcf:UniqueID	Shipment/ShipmentLines/ShipmentLine/ @OrderNo
_ord:OrderStatus/_ord:Status	S

WC path	Source
_ord:OrderItem/_ord:OrderItemIdentifier/_wcf:UniqueID	<ul style="list-style-type: none"> ► Shipment/ShipmentLines/ShipmentLine/@PrimeLineNo ► Shipment/ShipmentLines/ShipmentLine/@SubLineNo <p>If the “@SubLineNo” is not empty, append it to “@PrimeLineNo” with “-“ as a separator</p>
_ord:OrderItem/_ord:OrderItemStatus/_ord:Status	D



Integration implementation

This chapter contains the following sections:

- ▶ 8.1, “WebSphere Commerce DOM inventory cache management” on page 196
- ▶ 8.2, “WebSphere Enterprise Service Bus overview” on page 207
- ▶ 8.3, “WCToSSFSMediationModule: Processing of order capture” on page 214
- ▶ 8.4, “WebSphere Commerce DOM implementation” on page 223

8.1 WebSphere Commerce DOM inventory cache management

Certain backend inventory management systems, for example, legacy in-store systems, might have performance, scalability, and availability limitations. The inventory component and its services are designed with caching support to address these limitations. That is, the inventory availability of an item at a location can be cached by the component, either in memory or in the database. This results in the Distributed Order Management (DOM) inventory configuration and availability not being controlled by WebSphere Commerce Accelerator. Instead, WebSphere Commerce records the inventory information in the local database cache or memory cache, and the master data is contained in an external inventory system.

For a memory cache option, Distributed Object Cache is a feature of the WebSphere Application Server dynamic cache service. Distributed Object Cache holds Java objects for use in a distributed environment. For example, objects can be stored by one application server and then retrieved by other application servers in the same Data Replication Service (DRS) cluster. These cache instances are retrieved by a JNDI name that is configured on the Cache Instance resource. This cache can be configured so that objects are persistent, flushed to disk when the server is stopped, and loaded again upon restart.

The WebSphere Commerce DOM inventory cache is a custom caching solution, so that it can also cache inventory in the database, as well as in a WebSphere Application Server distributed object cache. For database caching, inventory availability is maintained in INVAVL, and for memory caching in the inventory distributed object cache. Note that a separate object cache instance is instantiated for caching inventory availability in memory, not to be confused with the cache instance used for JSP/command caching.

8.1.1 Inventory availability cache

This section contains details about DOM inventory availability caching support:

WebSphere Commerce can cache, at most, one inventory availability record for each combination of item and location. Each cached record contains the following information:

- ▶ Inventory status (for example, available, back-ordered, unavailable)
- ▶ Available quantity
- ▶ Availability date

- ▶ Availability offset

This is useful for delivery and buy-online-ship-to-store scenarios where the availability date is based on a lead time (offset) instead of an absolute date.

- ▶ Last update date

Commerce can cache an inventory availability record in memory or in the database, or both, depending on the inventory configuration of the item and location. A WebSphere distributed object cache is used to cache inventory availability records in memory.

Advantages of using a WebSphere distributed object cache include:

- ▶ It works in a clustered environment.
- ▶ It can be configured and managed using the WebSphere Cache Monitor.

Caching inventory availability records in memory is useful when it is either impossible or undesirable to batch-load the records into the database. For example, exporting all inventory availability records from a legacy in-store system could block inventory access for an unacceptably long period of time. Another example is if a large retailer has a large product assortment in combination with a sizable number of physical stores, it could be cost prohibitive to import all inventory availability records into the Commerce database.

DOM inventory caching configuration has the capability to capture global rules at the store/location level, or configure global rules for all stores. In addition, multiple rules can be applied to a single item, location, or store, and which takes precedence is done via a prioritization flag. This design allows for a retailer to apply default caching rules, as well as ones that might apply at the physical store level. Because catalog items can be grouped, caching rules can correlate to promotions.

Each inventory configuration contains the following options:

- ▶ Precedence
- ▶ Threshold quantity
- ▶ Expiry time of a cached record when available quantity \geq threshold quantity
- ▶ Expiry time of a cached record when available quantity $<$ threshold quantity
- ▶ Priority of a cached record
- ▶ Used by the distributed object cache to determine which cached records to purge in case of cache overflow
- ▶ Fallback values, used when the backend system is offline
- ▶ Additional options that control where the records should be cached (in memory or in the database, or both), whether real-time service requests

should be made to the backend system, and when the cached records should be decremented and updated

8.1.2 Caching examples

Certain eCommerce solutions might not require high inventory availability accuracy. For those stock items' thresholds, there is less of a need to invalidate cached records above the threshold and somewhat more frequently below the threshold.

There can be environmental constraints with the infrastructure of the in-store legacy systems where there is limited connectivity. In such a case, there might be no threshold at all, or cached records are never invalidated.

Sometimes real-time access to backend systems is disabled (for example, by a disaster) and the business decides that the shopping flow should not be interrupted. Commerce DOM inventory caching provides a capability to “dial up or down” the availability of inventory at various levels, which can decrease disruptions to the shopping flow.

Conversely, these scenarios require a high degree of caching. Configuration for items that require high inventory availability accuracy require:

- ▶ A high threshold
- ▶ Invalidated cached records somewhat frequently above the threshold
- ▶ Invalidated cached records very frequently below the threshold

8.1.3 Cache timing

Cached records are updated when:

- ▶ A cached record is refreshed by a real-time service request to the backend system when the cached record has expired.
- ▶ Cached records are decremented when an order is submitted for processing.
- ▶ Cached records are updated by batch-load and by SyncInventoryAvailability service requests.

Note: Cache updates can trigger JSP page cache invalidations.

Overt attempts to optimize DOM should *not* be a replacement of good store design. A good store design should balance performance and usability.

8.1.4 Inventory availability cache tables

Inventory configurations and related data are stored in the WebSphere Commerce database tables listed in Table 8-1.

Table 8-1 *Inventory availability cache tables*

Table name	Description
INVCNF	Represents DOM inventory cache configurations
INVCNFREL	Represents the relationship of an item and location with an inventory configuration
INVAVL	Represents an inventory availability record cached by the WebSphere Commerce database

The INVCNF and INVCNFREL tables hold the configuration information for the inventory cache and must be manually maintained. WebSphere Commerce maintains INVAVL for database caching requests.

Note: The WebSphere Commerce DOM inventory cache is a dedicated cache for inventory caching requirements and should not be confused with other Commerce caches.

8.1.5 Inventory cache schemas

This section discusses inventory cache sche.

INVCNF

Table 8-2 describes INVCNF schema.

Table 8-2 *INVCNF*

Column name	Columns type	Description
INVCNF_ID	BIGINT NOT NULL	Internal ID of this inventory configuration.
IDENTIFIER	VARCHAR(64) NOT NULL	External identifier of this inventory configuration.
DESCRIPTION	VARCHAR(254)	Administrative description of this inventory configuration.

Column name	Columns type	Description
PRECEDENCE	DOUBLE NOT NULL DEFAULT 0	Precedence of this inventory configuration when an item is associated with multiple inventory configurations at the location specified.
FLAGS	INTEGER NOT NULL DEFAULT 0	<p>A bit field for specifying the following inventory configuration options:</p> <ul style="list-style-type: none"> ▶ 0: The default value. None of the following flags are applicable. ▶ 1: Cache inventory availability in a WebSphere distributed object cache. ▶ 2: Cache inventory availability in the Commerce database. ▶ 4: Retrieve inventory availability from DOM by making an outbound GetInventoryAvailability service request when the information is not cached locally or when the cached information is no longer valid. ▶ 8: Decrement the cached inventory availability records when an order/shopping cart is submitted for processing. ▶ 16: Update the cached inventory availability records as the information is retrieved from DOM.
THRESHOLDQUANTITY	DOUBLE	The threshold quantity. It determines the expiry time used. A value of NULL is equivalent to a value of negative infinity, which means that the cached available quantity (minus the requested quantity if present) is always greater than or equal to the threshold quantity.

Column name	Columns type	Description
CHEXPABOVETHRES	INTEGER	The expiry time of a cached inventory availability record when the cached available quantity (minus the requested quantity if present) is greater than or equal to the threshold quantity. Such a record has expired when the number of seconds elapsed since the record was last updated is greater than this value. A value of 0 means that the record expires immediately, and a value of NULL means that the record never expires.
CHEXPBELOWTHRES	INTEGER	The expiry time of a cached inventory availability record when the cached available quantity (minus the requested quantity if present) is less than the threshold quantity. Such a record has expired when the number of seconds elapsed since the record was last updated is greater than this value. A value of 0 means that the record expires immediately, and a value of NULL means that the record never expires.
CACHEPRIORITY	INTEGER NOT NULL DEFAULT 0	The priority of a cached inventory availability record when cached in the WebSphere distributed object cache. Records with higher priorities will remain in the cache longer than those with lower priorities in the case of cache overflow.
FALLBACKINVSTAT	CHAR(4)	Fallback inventory status.
FALLBACKAVAILTIME	TIMESTAMP	Fallback availability date.
FALLBACKINVOFFSET	INTEGER	Fallback availability offset in seconds.
FIELD1	INTEGER	Customizable.
FIELD2	DECIMAL (20,5)	Customizable.

Column name	Columns type	Description
FI EL D3	VARCHAR(254)	Customizable.
QUANTITYMEASURE	CHAR(16) NOT NULL DEFAULT 'C62'	Unit of measure of the quantity above.
OPTCOUNTER	SMALLINT	Reserved IBM Internal Use.

INVCNFREL

Table 8-3 describes the INVCNFREL schema.

Table 8-3 INVCNFREL

Column name	Column type	Description
INVCNFREL_ID	BIGINT NOT NULL	Internal ID of this relationship.
INVCNF_ID	BIGINT NOT NULL	The inventory configuration ID.
CATENTRY_ID	BIGINT	The catalog entry ID. A value of NULL means that this relationship applies to all items at the location specified. A value of NULL for CATENTRY_ID, STORE_ID, and STLOC_ID means that this relationship applies to all items at all locations.
STORE_ID	INTEGER	The online store ID. Either this or the physical store ID can be specified, but not both. A value of NULL for both STORE_ID and STLOC_ID means that this relationship applies to the item at all locations. A value of NULL for CATENTRY_ID, STORE_ID, and STLOC_ID means that this relationship applies to all items at all locations.

Column name	Column type	Description
STLOC_ID	INTEGER	The physical store ID. Either this or the online store ID can be specified, but not both. A value of NULL for both STORE_ID and STLOC_ID means that this relationship applies to the item at all locations. A value of NULL for CATENTRY_ID, STORE_ID, and STLOC_ID means that this relationship applies to all items at all locations.
OPTCOUNTER	SMALLINT	Reserved IBM Internal Use.

8.1.6 INVCNF configuration and test results

Each row of INVCNF represents a DOM inventory cache configuration. WebSphere Commerce InfoCenter contains the table schema for INVCNF:

<http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.database.doc/database/invcnf.htm>

The INVCNF.FLAGS description displays the various bit values that can be part of the final bit calculation, but it does not provide all of the permutations that would go into the FLAGS field. For example, consider the following cache configuration:

- ▶ 2: Cache in database.
- ▶ 8: Decrement the cached inventory availability records when an order/shopping cart is submitted for processing.
- ▶ 16: Update the cached inventory availability records as the information is retrieved from DOM.

The bit calculation would be 26 (2+8+16).

Table 8-4 shows all likely bitwise permutations for FLAGS.

Table 8-4 INVCNF FLAGS bit logic

INVCNF_ID	IDENTIFIER	DESCRIPTION	No cache	Mem cache	DB cache	Inv availability	Decr.	Update	FLAGS
10001	invcnf1	Cache in database, decrement, update			2		8	16	26
10002	invcnf2	Cache in memory, decrement, update		1			8	16	25
10003	invcnf3	No cache, decrement, update					8	16	24
10004	invcnf4	Cache in database, decrement			2		8		10
10005	invcnf5	Cache in memory, decrement		1			8		9
10006	invcnf6	No cache, decrement					8		8
10007	invcnf7	Cache in database, update			2			16	18
10008	invcnf8	Cache in memory, update		1				16	17
10009	invcnf9	No cache, update						16	16
10010	invcnf10	Cache in database			2				2
10011	invcnf11	Cache in memory		1					1
10012	invcnf12	No cache	0						0
10013	invcnf13	Cache in database, decrement, update			2	4	8	16	30

Note: The Madison INVCNF FLAGS values should be updated to reflect those values found in Table 8-4.

In the ITSO lab environment based on the Madison's store, tests were ran on several of the likely scenarios for FLAGS settings so as to validate the behavior of DOM inventory availability caching. Table 8-5 lists the findings.

Table 8-5 INVCNF FLAGS findings

INVCNF FLAGS	INVCNF THRESHOLD	Cache memory	Cache table	On hand (catalog browse)	Inventory availability (cart)	Allocate inventory (checkout)
26	100		Yes	In Stock	N	Y
25	100	Yes		In Stock	N	Y
10	100		Yes	In Stock	N	Y
9	100	Yes		In Stock	N	Y
8	100			Out of Stock	N	Y
18	100		Yes	In Stock	N	Y
17	100	Yes		In Stock	N	Y
16	100			Out of Stock	N	Y
2	100		Yes	In Stock	N	Y
1	100	Yes		In Stock	N	Y
0	100			Out of Stock	N	Y
30	100			In Stock	N	Y
4	100			In Stock	Y	Y

As our focus was on confirming the behavior of INVCNF.FLAGS, we wanted to keep constants in the other configuration fields so as to minimize their impact to the results. The remaining fields were set to these values:

- ▶ PRECEDENCE = 1
- ▶ THRESHOLDQUANTITY = 100
- ▶ CHEXPABOVETHRES = NULL
- ▶ CHEXPBELOWTHRES = NULL
- ▶ CACHEPRIORITY = 0
- ▶ FALLBACKINVSTAT = "UAVL"
- ▶ FALLBACKAVAILTIME = NULL
- ▶ FALLBACKINVOFFSET = NULL

Captured behaviors for INVCNF FLAGS settings are:

- ▶ Database caching (2)

Any permutation of the bit calculation that arrives at FLAGS=26, 10, 18, or 2, will update INVAVL, which is the table that Commerce uses for database caching. This table is updated only after `reserveAvailableInventory` is called after cart is confirmed (billing method is approved). Database and memory cache are mutually exclusive

- ▶ Memory caching (1)

Any permutation of the bit calculation that arrives at FLAGS=25, 9, 17, or 1 will maintain the inventory in the distributed object inventory cache. This cache is updated only after `reserveAvailableInventory` is called after the cart is confirmed (the billing method is approved). Database and memory cache are mutually exclusive.

- ▶ No cache (8, 16, 0)

Any *no cache* configuration setting will create an out-of-stock condition, as neither database nor memory cache are being used. As the retrieve inventory availability setting (4) is not set, there is no inventory to present to the page, so an “Out of Stock” message is presented to the shopper. The shopping flow, however, is *not* interrupted.

- ▶ Retrieve inventory availability (4)

When set, DOM makes a request to Sterling Order Management for inventory availability as early as catalog browsing, and again when checkout is chosen. As this call obtains real-time inventory, any other permutation will disable it. Thus, this flag is mutually exclusive to all other configuration flags.

Based on the above findings, one could consider the following as the first choice for a FLAGS setting based on the following conditions. Of course, you might arrive at different results based on requirements and constraints and by altering the other settings in INVCNF.

- ▶ If there are no constraints to accessing Sterling managed inventory, possibly the most accurate inventory would be attained with FLAGS=4, and no local caching is maintained.
- ▶ If there are backend constraints and caching is required, an optimum setting would be FLAGS=26 for database caching and FLAGS=25 for distributed object caching. Most Commerce architects would recommend distributed object caching for performance reasons, and overall it should be more robust, as this setting can fully leverage WebSphere DynaCache clustering.

8.1.7 DOM interfaces

Table 8-6 and Table 8-7 detail inbound service interfaces used by the external DOM.

Table 8-6 Inbound service interfaces used by DOM integration

Component	BOD	Action code/access profile	Description
Inventory	SyncInventoryAvailability	Change	Updates the cached inventory availability records
Order	SyncOrder	Change	Updates the order fulfillment records
Order	ProcessOrder	Cancel	Cancels an order

Table 8-7 Outbound service interfaces used by DOM integration

Component	BOD	Action code/access profile	Description
ExternallInventory	GetInventoryAvailability	IBM_Store_Details	Retrieves the inventory availability of items at the specified locations
ExternallInventory	ProcessInventoryRequirement	ReserveInventory	Reserves inventory for order items in an order or shopping cart
ExternallInventory	ProcessInventoryRequirement	CancelInventoryReservation	Cancels the inventory reservations of order items in an order or shopping cart
ExternalOrder	ProcessOrder	Transfer	Transfers an order for further processing

8.2 WebSphere Enterprise Service Bus overview

In this section, the implementation techniques in WebSphere Enterprise Service Bus (WESB) are discussed as they would relate to the solution provided with the release of WebSphere Commerce Version 7 Feature Pack 2.

8.2.1 WebSphere Enterprise Service Bus key terms

Table 8-8 summarizes the key terms in the context of WebSphere Enterprise Service Bus (WESB) that are discussed in this section.

Table 8-8 WESB key terms

Term	Explanation
Mediation	A service request interception by an ESB that typically centralizes logic such as routing, transformation, and data handling
Mediation module	The basic building block in WebSphere Enterprise Service Bus for creating mediations
Export	Exposes the interfaces of an mediation module and contains the bindings
Stand-alone reference	The external publishing of an interface for SCA clients only (without a WSDL description)
Import	Represents the service providers that are invoked by a mediation module
Binding	The protocols and transports that are assigned to exports and imports
Mediation flow component	The container for mediation logic inside a mediation module that provides interfaces and that uses references
Interface	Define access points that are defined using WSDL
Operation	Represent interactions that can be 1-way (only input parameters) and 2-way (input and output parameters)
Partner reference	The declaration of the referenced interfaces of an mediation flow component
Wire	An association between components inside a mediation module and exports/imports/stand-alone references
Mediation flow	The processing steps that are defined for each interface in the form of a request flow and usually a response flow
Mediation primitive	Units of message processing inside a mediation flow that provide different terminals
Service message object (SMO)	A data object that represents the context, content, and header information of an application message that is created during a mediation flow

8.2.2 Mediations, service consumers, and service providers

A service interaction in a service-oriented architecture (SOA) defines both service consumers and service providers. The role of WebSphere Enterprise Service Bus is to intercept the requests of service consumers and fulfill additional

tasks in mediations in order to support loose coupling. When the mediation completes, the relevant service providers should be invoked. The mediation tasks include:

- ▶ Centralizing the routing logic so that service providers can be exchanged transparently
- ▶ Performing tasks such as protocol translation and transport mapping
- ▶ Acting as a facade in order to provide different interfaces between service consumers and providers
- ▶ Adding logic to provide tasks such as logging

See Figure 8-1.

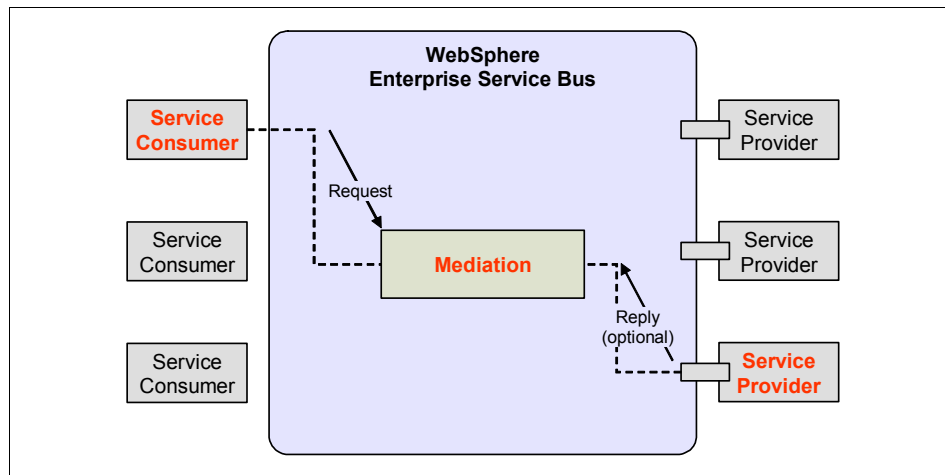


Figure 8-1 Service mediation

WebSphere Enterprise Service Bus can interconnect a variety of service consumers and providers using standard protocols including:

- ▶ JMS
- ▶ SOAP over HTTP (for web services)
- ▶ SOAP over JMS (for web services)

WebSphere Enterprise Service Bus supports diverse messaging interaction models to meet your requirements, including the following models:

- ▶ One-way interactions
- ▶ Request-reply
- ▶ Publish/subscribe

8.2.3 Mediation modules

The mediation module is a Service Component Architecture (SCA) component that can process or mediate service interactions. As illustrated in Figure 8-2, the mediation module is externalized or made available through an export, which specifies the interfaces that are exposed. These interfaces are defined in a WSDL document.

The mediation module typically invokes other service providers. These providers are declared with the creation of an import, which represents an external service to be invoked (Figure 8-2).

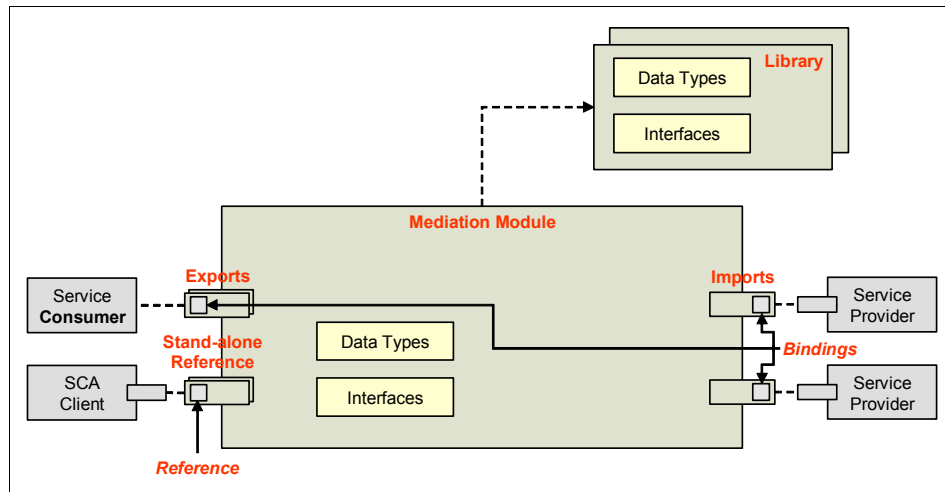


Figure 8-2 WESB mediation module

For each export and import, an interface needs to be specified. Each interface has multiple operations, which in turn can have multiple input and output parameters that are associated with either simple data types or business objects. A 1-way operation has only input parameters.

8.2.4 Mediation flow components

Inside a mediation module there can be one mediation flow component. Mediation flow components offer one or more interfaces and use one or more partner references. Both get resolved, assigning them to exports or imports via wires (Figure 8-3).

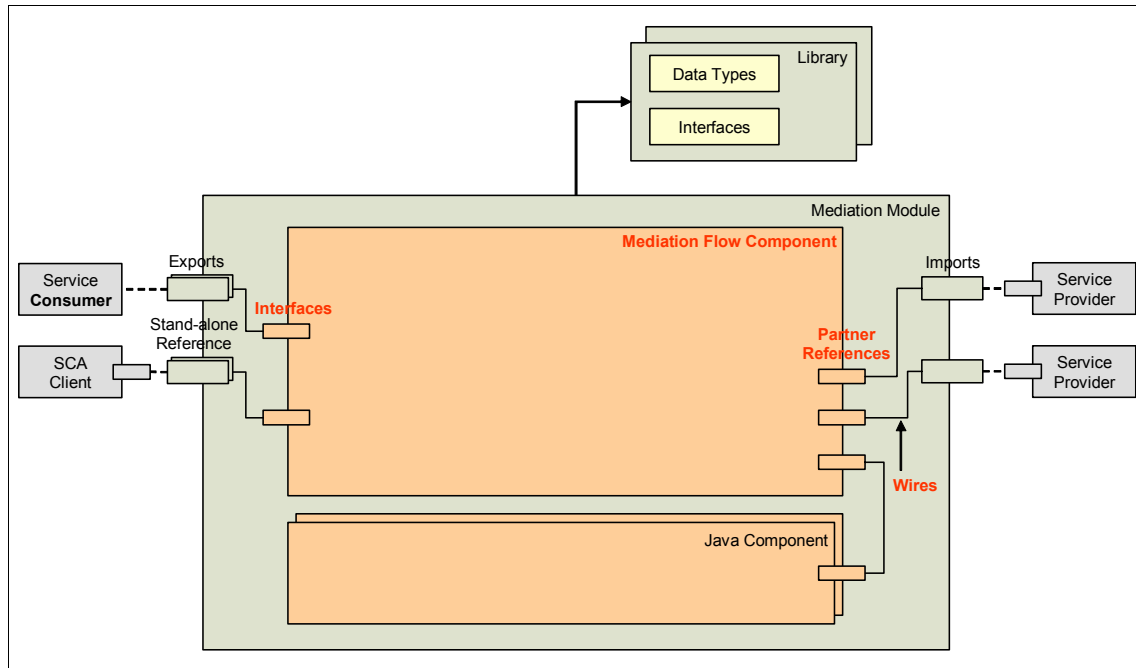


Figure 8-3 Mediation flow components

8.2.5 Mediation flows

Mediation flows contain the high-level mediation logic. Thus, the different processing steps of a request are declared in a graphical way. In WebSphere Enterprise Service Bus, the processing of requests is separated from processing of responses. Therefore, we distinguish between a request flow and a response flow. In both directions, logic can be added or modifications can be applied.

Note: Mediation flows need to be defined for every operation that gets exposed via an export of a mediation module (Figure 8-4 on page 212).

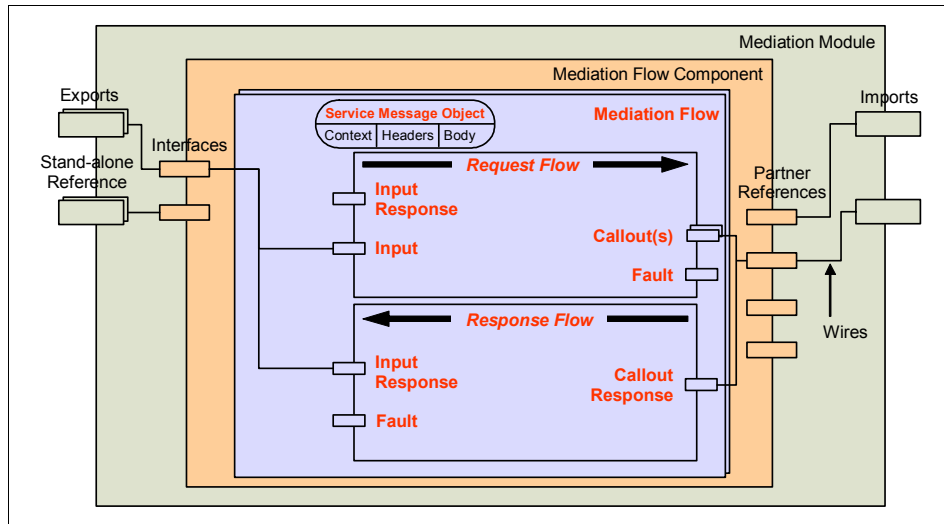


Figure 8-4 Mediation flows

Mediation flows consist of a sequence of processing steps that are executed when an input message is received. A request flow begins with a single input for the source operation and can have multiple callouts. If a message is to be returned to the source directly after processing, it can be wired to an input response in the request flow. If fault messages are defined in the source operation, an input fault is also created.

A response flow begins with one or more callout responses and ends with a single input response (and optionally a callout fault). Both a request flow and a response flow are associated with a mediation flow. The request flow can map data to a correlation context and the transient context.

In terms of the actual data, WebSphere Enterprise Service Bus introduces the service message object (SMO). SMO is a special kind of a service data object that represents the content of an application message as it passes through a mediation flow component. In addition to the payload in the body, it contains context and header information, which can be accessed and acted upon inside the mediation flows.

8.2.6 Mediation primitives

Mediation primitives (Figure 8-5) are the smallest building blocks in WebSphere Enterprise Service Bus. They are wired and configured inside mediation flows. They let you:

- Change the format, content, or target of service requests
- Log messages.
- Do database lookups.
- And so forth.

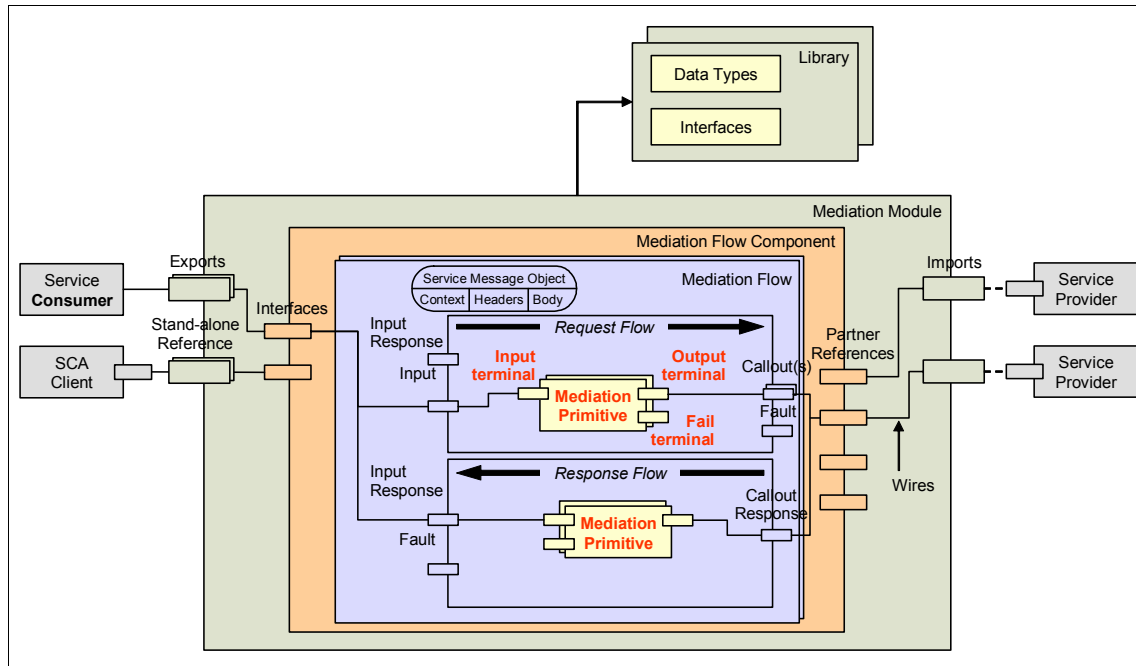


Figure 8-5 Mediation primitives

WebSphere Integration Developer and WebSphere Enterprise Service Bus provide the following standard mediation primitives:

- The **MessageLogger** primitive logs a copy of a message to a database for future retrieval or audit. The integration developer can customize the primitive by, for example, naming the database.
- The **DatabaseLookup** primitive retrieves values from a database to add them to a message.
- The **MessageFilter** primitive compares the content of a message to expressions configured by the developer and routes the message to the next mediation primitive based on the result.

- ▶ The XSLT primitive transforms messages according to transformations defined by an XSL style sheet.
- ▶ The stop primitive silently terminates the path through the mediation flow.
- ▶ The fail primitive throws an exception and terminates the path through the mediation flow.
- ▶ The custom mediation primitive allows the user to implement her own mediate method using Java. The custom mediation, like the other primitives, receives a Service Message Object and returns a Service Message Object. It can be used to perform tasks that cannot be performed by using the other mediation primitives.

8.3 WCToSSFSMediationModule: Processing of order capture

This section reviews the out-of-the-box order processing mediation flow named WCToSSFSMediationModule, viewed from WebSphere Integration Developer (WID) and deployed on WebSphere Enterprise Service Bus (WESB). This section assumes that the reader already has prior experience working with WESB/WID or has read the previous introductory section, 8.2, “WebSphere Enterprise Service Bus overview” on page 207. If you need further preparation, see “Related publications” on page 243.

8.3.1 WCToSSFSMediationModule mediation module

The out-of-the-box mediation module for SSFS-to-WebSphere integration is named WCToSSFSMediationModule. It might be located in WID (navigate to **Projects** → **WCToSSFSMediationModule** and double-click **Assembly Diagram**). See Figure 8-6 for the WCToSSFSMediationModule project location.

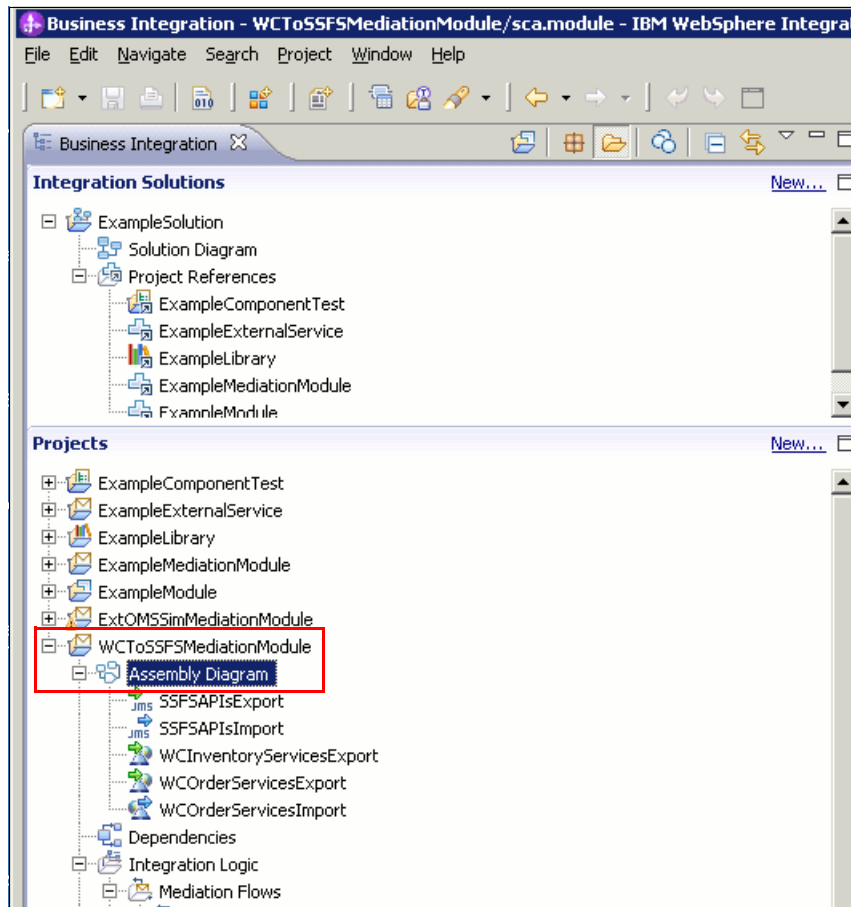


Figure 8-6 WCToSSFSMediationModule project location

The WCTToSSFSMediationModule assembly diagram then is displayed in the Editor pane (Figure 8-7).

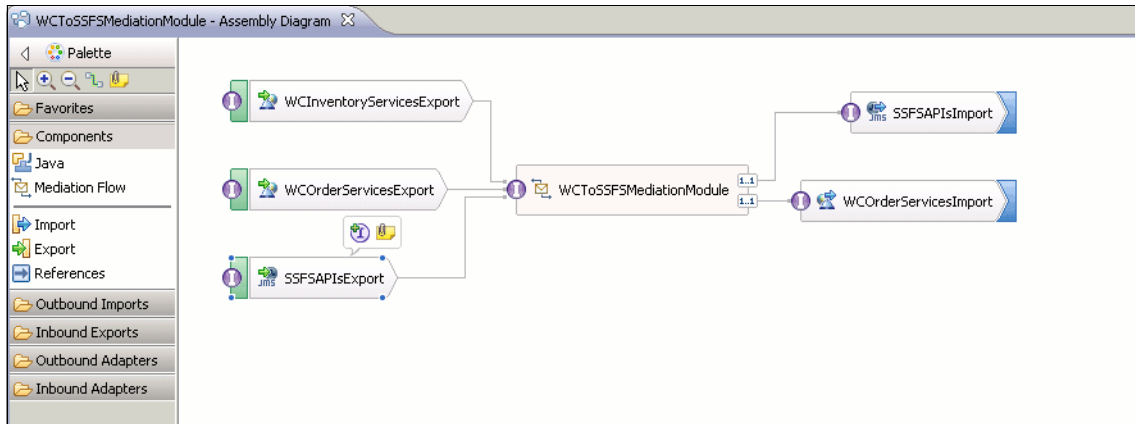


Figure 8-7 WCTToSSFSMediationModule assembly diagram

WCTToSSFSMediationModule is used for multiple service requests. It is a design choice that leverages shared integration components. This module is built for easy extension of un-implemented integration flows. Shared components sometimes have efficiency/performance downsides, though there is nothing in this design suggesting such. Regardless, the design can be easily revisited if there are such concerns. The WESB technology is very adaptable.

Double-clicking the export and import icons in the WID editor view displays the operations that can be viewed by navigating to **Properties** → **Details** (Figure 8-8).

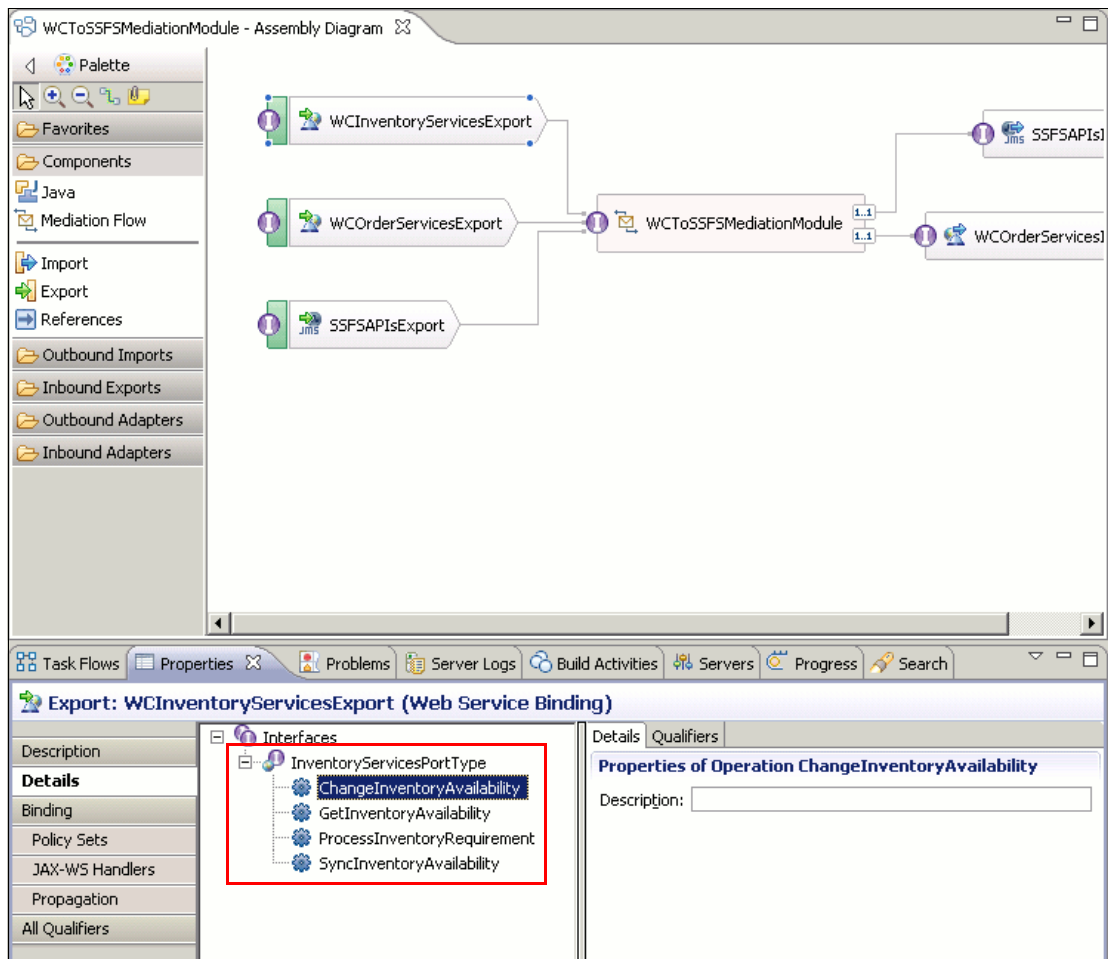


Figure 8-8 WCToSSFSMediationModule export operations

The WCToSSFSMediationModule exported interfaces and imported service providers are listed with their specific operations in Table 8-9. As noted in Table 8-9, not all interfaces are implemented as of WebSphere Commerce V7 Feature Pack 2. This overview focuses on the WOrderServicesExport export interface. This requester is a WebSphere Commerce web client requesting that the provider (SSFS) accept the order.

Table 8-9 WCToSSFSMediationModule operations

Export interface	Client	Binding	Export operations	V7 FEP2	Import operations
WCInventoryServicesExport	Commerce	Web service	ChangeInventoryAvailability	No	
			GetInventoryAvailability	Yes	findInventory
			ProcessInventoryRequirement	Yes	reserveAvailabl eInventory
			SyncInventoryAvailability	No	
WOrderServicesExport	Commerce	Web service	ChangeOrder	No	
			ChangeQuote	No	
			GetOrder	No	
			SyncQuote	No	
SSFSAPIsExport	Sterling	JMS	SendOrderDetails	Yes	syncOrder
			SendShipmentDetails	Yes	syncOrder

8.3.2 WCToSSFSMediationModule mediation flow

Navigate to the mediation component overview by double-clicking the WCToSSFSMediationModule icon in the WCToSSFSMediationModule assembly diagram. The mediation component overview lists the operations and maps the relationships between the exports and imports ports (Figure 8-9). Note that the client (export interfaces) can be either SSFS or WebSphere Commerce based on the requirements.

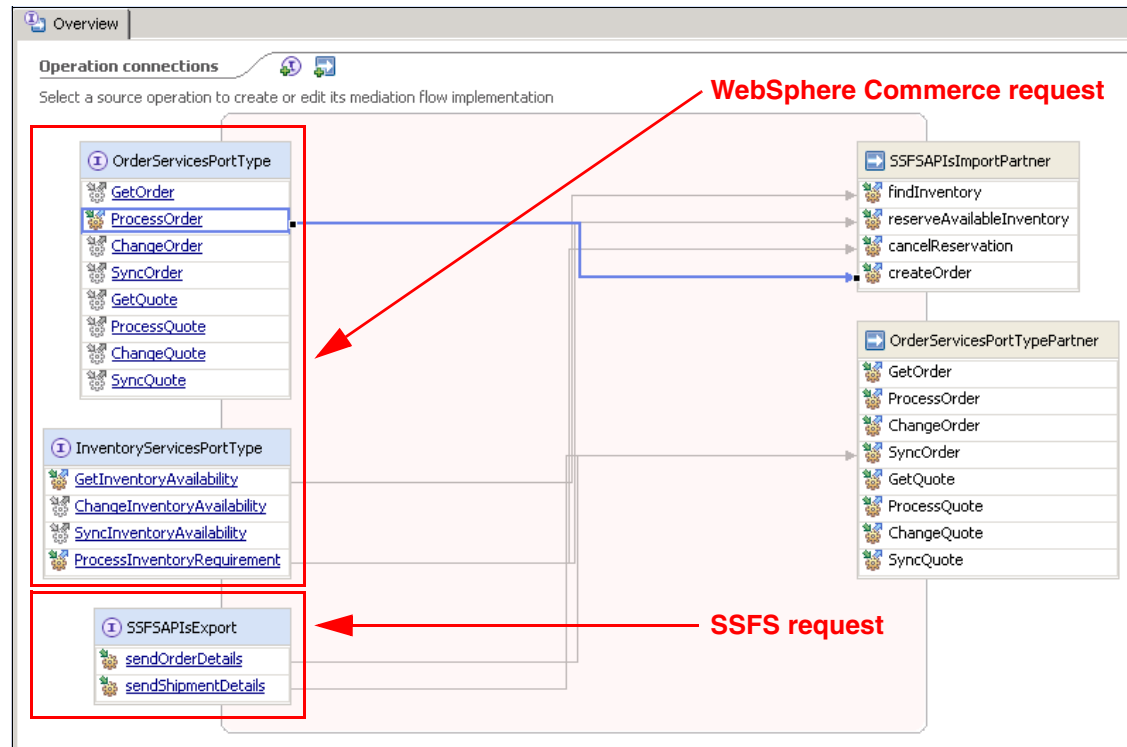


Figure 8-9 WCToSSFSMediationModule component overview

Every export and import has to be associated with a binding. A binding identifies a specific type of invocation for a service consumer or provider. WebSphere Enterprise Service Bus supports several bindings. Those used by WCToSSFSMediationModule are:

- ▶ SSFS - JMS
- ▶ WebSphere Commerce - web services using SOAP/HTTP

For all flows (scenarios) detailed in this book, WebSphere Enterprise Service Bus is acting as a bridge between the bindings. The developer has minimal duties other than basic configuration bridging between these two technologies.

We now look a little closer at the mediation module. Navigate to the WCToSSFSMediationModule mediation flow by double-clicking the WCToSSFSMediationModule icon in the WCToSSFSMediationModule assembly diagram. Click the **Overview** tab, which displays all of the mapped operations belonging to the module. Highlighting any of the solid lines, each representing one flow, emphasizes that particular flow by placing highlighted emphasis on that flow. Click the **ProcessOrder** operation (Figure 8-10). This brings the focus to the operation with both request and response flows.

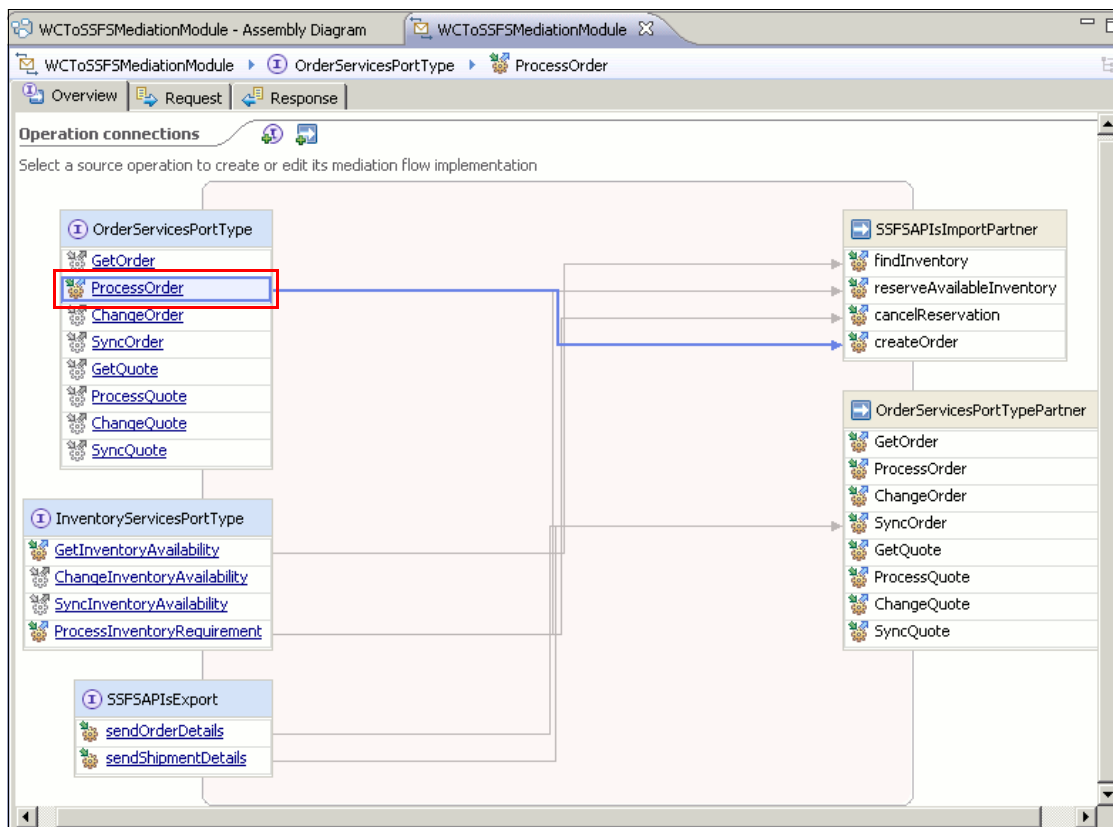


Figure 8-10 WCToSSFSMediationModule mediation flow overview

8.3.3 WCToSSFSMediationModule mediation flow request

For the WCToSSFSMediationModule mediation flow request, click the **Request** tab to view the ProcessOrder request flow shown in Figure 8-11.

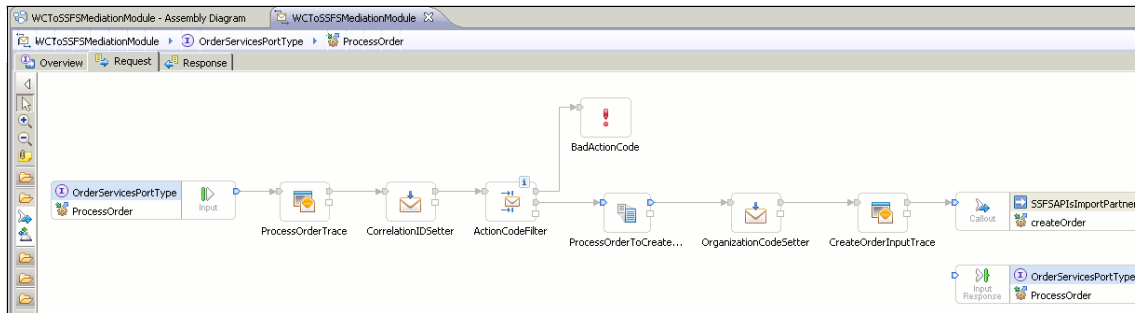


Figure 8-11 WCToSSFSMediationModule mediation flow request

The WCToSSFSMediationModule ProcessOrder request flow comprises fundamental building blocks called primitives. Every request and response for every mediation requirement has a similar work flow as ProcessOrder, changed only by the requirements. Each is uniquely identified by a message type so that the module can correctly channel the desired message through the matching flow, as explained below.

8.3.4 WCToSSFSMediationModule mediation flow request primitives

The ProcessOrder mediation flow primitives that warrant description are detailed here:

- ProcessOrderTrace traces the metadata and message in xml format to the local server log (WebSphere Application Server SystemOut.log) (Figure 8-12).

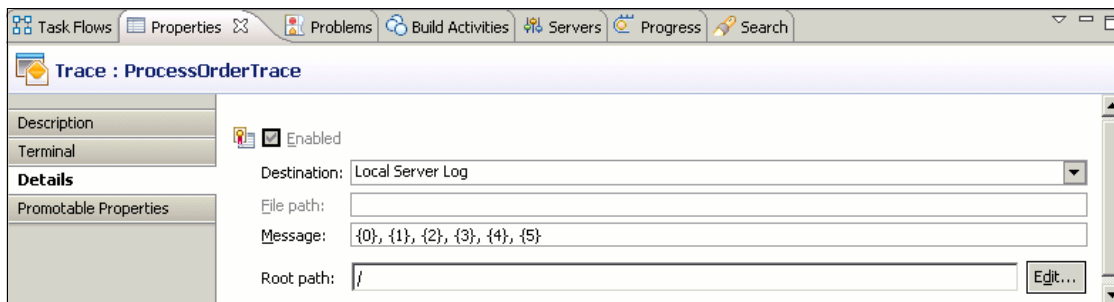


Figure 8-12 ProcessOrderTrace

- CorrelationIDSetter captures an ID used for JMS correlation and copies /body/ProcessOrder/ApplicationArea/BODID/value to /headers/JMSHeader/JMSCorrelationID (Figure 8-13).

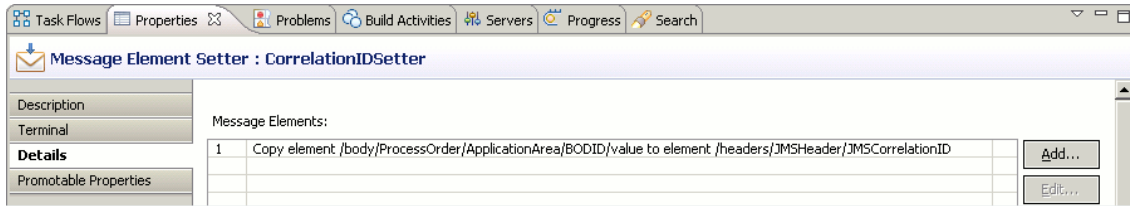


Figure 8-13 CorrelationIDSetter

- ActionCodeFilter: As many message types are routed through this mediation flow, we need a way to filter for the desired message type. This primitive conditionally routes the message based on the message content, by comparing the message against a list of predefined expressions. In this case the value that will allow the rest of this flow to continue is TransferOrder. Per this rule
/body/ProcessOrder/DataArea/Process/ActionCriteria[1]/ActionExpression[1]/@actionCode = 'TransferOrder' (Figure 8-14).

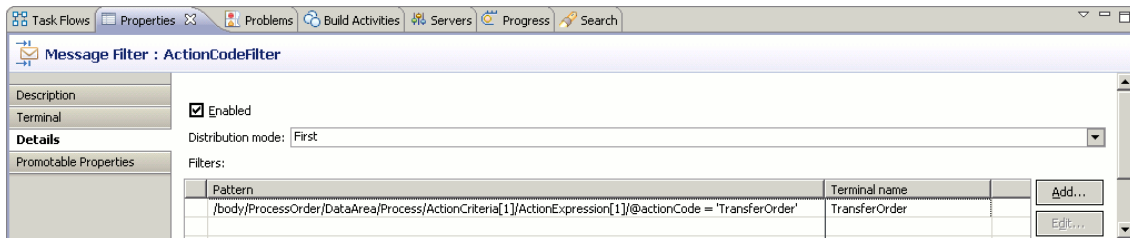


Figure 8-14 ActionCodeFilter

- ProcessOrderToCreateOrderInput: XSLT primitive that maps the message to the desired format expected by the SSFS order system. This primitive references a custom stylesheet captured in Appendix A, “Supporting content” on page 233.

8.3.5 WCToSSFSMediationModule mediation flow response

In the mediation flow, click the **Response** tab to view the ProcessOrder response flow (Figure 8-15). The reader can apply the analysis provided in the WCToSSFSMediationModule mediation flow request to this section.

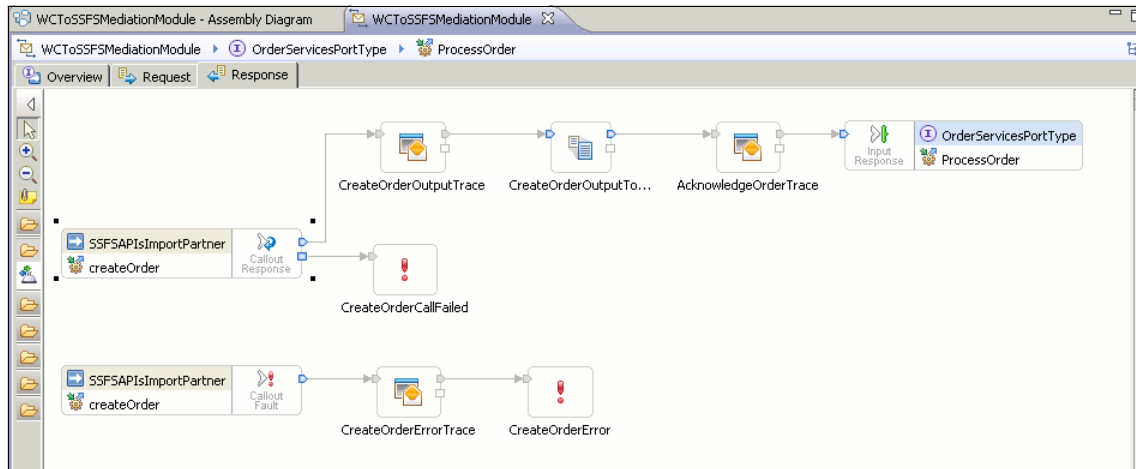


Figure 8-15 WCToSSFSMediationModule mediation flow response

Building a simple, yet functional, WebSphere Enterprise Service Bus (WESB) mediation module for DOM integration using WID introduces you to several key concepts. These concepts include working with WID as a development platform, working with mediation flows, and using XSL transformation primitives in mediation flows to transform WebSphere Commerce OAGIS BOD messages to and from messages in the backend system's format.

8.4 WebSphere Commerce DOM implementation

This section discusses the relevant implementation for WebSphere Commerce inventory/order/store DOM implementation.

8.4.1 WebSphere Commerce commands behavior with business rules

The sections are a synopsis of the WebSphere Commerce commands for inventory/order/store DOM implementation, which have business rules worth highlighting.

FetchInventoryAvailabilityCmdImpl

If the inventory system of the store is DOM:

- ▶ It retrieves the inventory configuration applicable to each item/location combination.
- ▶ For each combination, it:
 - Retrieves the inventory availability record of this combination cached by the WebSphere distributed object cache if this is enabled by the inventory configuration.

Note: Consider caching inventory configurations and their relationships with items and locations in memory, given that they are frequently accessed but seldom updated.

- Retrieves the inventory availability record of this combination cached by the Commerce database if this is enabled by the inventory configuration, and that the information is not already cached by the WebSphere distributed object cache.
- ▶ For combinations that do not have their inventory availability records cached locally, or that the cached information has expired:
 - Retrieves the inventory availability records of these combinations from DOM by making an outbound GetInventoryAvailability service request using the ExternalInventory component.
 - Updates the inventory availability record cached by the WebSphere distributed object cache if this is enabled by the inventory configuration.
 - Updates the inventory availability record cached by the Commerce database if this is enabled by the inventory configuration.
- ▶ For each combination that inventory availability is still unknown, it composes the service response using fallback values specified in the inventory configuration.

Table 8-10 lists possible results.

Table 8-10 Possible inventory outcomes

Description	Inventory status	Available quantity	Availability date
The item is available and the available quantity is known.	Available	(The available quantity)	<= current time
The item is available but the available quantity is unknown or unlimited.	Available	Null	<= current time
The item is unavailable but is back-orderable, and the availability date is known.	Backorderable	Null	(The availability date)
The item is unavailable.	Unavailable	Null	Null

ValidatePaymentMethodCmdImpl (Update)

This is the same as the default behavior, except the payment policy is valid if:

- ▶ Its name is PayInStore, and the order does not contain any delivery order item or existing payment instruction.
- ▶ Its name is not PayInStore, and the order does not contain any existing payment instruction that references the PayInStore payment policy.

DoInventoryActionCmdImpl

This is the same as the default behavior, except if the inventory system of the store is DOM (that is, STORE.INVENTORYSYSTEM=-5):

1. If the caller command is **OrderItemBaseCmd** or **OrderItemDisplayCmd**, and the action is **DoInventoryActionCmd.GET_DEFAULT_ATP_PARAMETER**:
 - a. If the order is locked:
 - It composes the InventoryRequirement SDO by calling **ComposeOrderCmd** with access profile IBM_Inventory
 - It sets the transaction cache custom property **Inventory.cancelInventoryReservation.<order ID>** to the InventoryRequirement SDO.

- b. It sets the transaction cache custom property `Inventory.lastUpdateDate.<order ID>` to the last update date of the order.
- 2. If the caller command is **OrderItemBaseCmd**, the action is `DoInventoryActionCmd.UPDATE_FULFILLMENTCENTER`, and the order is unlocked:
 - a. It removes the `InventoryRequirement` SDO assigned to the transaction cache custom property `Inventory.cancelInventoryReservation.<order ID>`.
 - b. It calls the inventory component's `ProcessInventoryRequirement` service with action code `CancelInventoryReservation` to cancel inventory reservation if the transaction cache custom property was set.
 - c. It composes another `InventoryRequirement` SDO by calling **ComposeOrderCmd** with access to profile `IBM_Inventory`.
 - d. It calls the inventory component's `ProcessInventoryRequirement` service with action code `CheckInventory` to check inventory.
- 3. If the caller command is **OrderItemDisplayCmd**, the action is `DoInventoryActionCmd.CHECK_INVENTORY`, and the order is unlocked:
 - a. It removes the `InventoryRequirement` SDO assigned to the transaction cache custom property `Inventory.cancelInventoryReservation.<order ID>`.
 - b. It calls the Inventory component's `ProcessInventoryRequirement` service with action code `CancelInventoryReservation` to cancel inventory reservation if the transaction cache custom property was set.
 - c. It composes another `InventoryRequirement` SDO by calling **ComposeOrderCmd** with access to profile `IBM_Inventory`.
 - d. It removes the date assigned to the transaction cache custom property `Inventory.lastUpdateDate.<order ID>`.
 - e. It calls the Inventory component's `ProcessInventoryRequirement` service with action code `CheckInventory` to check inventory if the number of seconds elapsed since the date is greater than the `STORE.QUOTEGOODFOR` value.
- 4. If the caller command is **OrderPrepareCmd** and the action is `DoInventoryActionCmd.CHECK_INVENTORY`:
 - a. It composes the `InventoryRequirement` SDO by calling **ComposeOrderCmd** with access to profile `IBM_Inventory`.
 - b. It calls the inventory component's `ProcessInventoryRequirement` service with action code `ReserveInventory`.
- 5. It updates the fulfillment information of the order items, including fulfillment center ID, inventory status, and availability date, using information from the service response.

Refer to Table 8-11 for possible order item fulfillment information.

Table 8-11 *ValidatePaymentMethodCmdImpl (Update)*

Description	Inventory status	Availability date
The order item can be fulfilled immediately, but inventory is not allocated yet.	DOM: AVL. Other inventory systems: NALC	<= current time
The order item cannot be fulfilled immediately but is backorderable.	DOM: BA. Other inventory systems: NALC	(The availability date)
The order item cannot be fulfilled.	DOM: UAVL. Other inventory systems: NALC	null
The order item can be fulfilled immediately and inventory has already been allocated.	ALLC	<= current time
The order item cannot be fulfilled immediately and is backordered.	BO	(The availability date)

ProcessInventoryRequirementCheckInventoryActionCmdImpl:

- Retrieves the inventory configuration applicable to each item/location combination in the order/shopping cart.
For each combination, it:
 - Retrieves the inventory availability record of this combination cached by the WebSphere distributed object cache if enabled by the inventory configuration.
 - Retrieves the inventory availability record of this combination cached by the Commerce database if enabled by the inventory configuration and the information is not already cached by the WebSphere distributed object cache.
- For combinations that do not have inventory availability cached locally, or whose cached information has expired, it:
 - Retrieves the inventory availability records of these combinations from DOM by making an outbound GetInventoryAvailability service request using the ExternalInventory component.

- Updates the inventory availability records cached by the WebSphere distributed object cache if this is enabled by the inventory configuration.
- Updates the inventory availability records cached by the Commerce database if this is enabled by the inventory configuration.
- Composes the service response. For each combination, set the fulfillment information of the corresponding order items to:
 - Inventory availability of the combination if the available quantity is greater than or equal to the requested quantity.
 - The fallback values specified in the inventory configuration if the available quantity is unknown or less than the requested quantity.

ProcessInventoryRequirementReserveInventoryActionCmdImpl

This makes an outbound ProcessInventoryRequirement service request to DOM with action code ReserveInventory using the ExternalInventory component.

ProcessInventoryRequirementCancelInventoryReservationActionCmdImpl

This makes an outbound ProcessInventoryRequirement service request to DOM with action code CancelInventoryReservation using the ExternalInventory component.

ProcessInventoryRequirementDecrementCacheActionCmdImpl

This:

1. Retrieves the inventory configuration applicable for each item/location combination.
2. For each combination, it:
 - Retrieves the inventory availability record of this combination cached by the WebSphere distributed object cache if this is enabled by the inventory configuration.
 - Retrieves the inventory availability record of this combination cached by the Commerce database if this is enabled by the inventory configuration.
 - Decrements the available quantity of the records that the inventory status is available by the requested quantity. Sets the inventory status of the records to the fallback inventory status if the availability quantity is decremented to 0.

ComposeOrderDetailsCmdImpl

This is the same as default behavior except for the exceptions listed in Table 8-12.

Table 8-12 *ComposeOrderDetailsCmdImpl*

SDO	Behavior
/Order/OrderItem[n]/OrderItemShippingInfo/PhysicalStoreIdentifier/UniqueID	Unique ID of the pickup location if the shipping mode is PickupInStore (that is, its code is PickupInStore and its carrier is NULL)
/Order/OrderItem[n]/OrderItemShippingInfo/PhysicalStoreIdentifier/ExternalIdentifier	External identifier of the pickup location if the shipping mode is PickupInStore
/Order/OrderItem[n]/OrderItemShippingInfo/ShippingMode/ShippingModelIdentifier/ExternalIdentifier/StoreUniqueID	Store ID of the shipping mode
/Order/OrderItem[n]/OrderItemShippingInfo/ShippingMode/ShippingModelIdentifier/ExternalIdentifier/ShippingModeCode	Code of the shipping mode
/Order/OrderItem[n]/OrderItemShippingInfo/ShippingMode/ShippingModelIdentifier/ExternalIdentifier/Carrier	Carrier of the shipping mode

ComposeUsableShippingInformationCmdImpl

The is the same as the existing behavior, except for the addition to the order SDO listed in Table 8-13.

Table 8-13 *ComposeUsableShippingInformationCmdImpl*

SDO	Behavior
/Order/OrderItem[n]/UsableShippingMode[m]/ShippingModelIdentifier/ExternalIdentifier/StoreUniqueID	Store ID of the shipping mode.
/Order/OrderItem[n]/UsableShippingMode[m]/ShippingModelIdentifier/ExternalIdentifier/ShippingModeCode	Code of the shipping mode.
/Order/OrderItem[n]/UsableShippingMode[m]/ShippingModelIdentifier/ExternalIdentifier/Carrier	Carrier of the shipping mode.

ProcessOrderSubmitEventCmdImpl

This is the same as the existing behavior, except if the inventory system of the store is DOM (that is, `STORE.INVENTORYSYSTEM=-5`), then it:

- ▶ Uses the order component's `GetOrder` service to retrieve the order details, then makes an outbound `ProcessOrder` service request with the action code `process` using the `ExternalOrder` component.
- ▶ Calls the inventory component's `ProcessInventoryRequirement` service with action code `DecrementCache`.

ChangeInventoryAvailabilityCmdImpl

This:

- ▶ Uses the generated code to update the inventory availability record cached by the Commerce database (or creates one) if this is enabled by the inventory configuration.
- ▶ Updates the inventory availability record cached by the WebSphere distributed object cache (or creates one) if this is enabled by the inventory configuration.

OrderStatusCmdImpl

This is the same as the existing behavior, except if the inventory system of the store is DOM (that is, `STORE.INVENTORYSYSTEM=-5`), then it reserves and finalizes payment for order items whose fulfillment statuses have just been updated to D (deposited).

OrderStatusNotifySendCmdImpl

This is the same as the existing behavior, except if the inventory system of the store is DOM (that is, `STORE.INVENTORYSYSTEM=-5`), then it:

1. Identifies the order items whose fulfillment statuses have just been updated.
2. Groups the order items above by fulfillment status.
3. Sends a notification e-mail to the order owner for each group above using the message type configuration `OrderStatusNotify_<fulfillment status>` if it has been defined.

OrderItemBaseCmdImpl

This is the same as the existing behavior, except that it:

- ▶ Sets the fulfillment center ID of the order item to the fulfillment center corresponding to the physical store if the shipping mode of an order item is PickupInStore and the physicalStoreId_i parameter is specified.
- ▶ Recalculates the calculation usages specified by the calculationUsageId parameter if the calculateOrder parameter is set to 1. The calculationUsageId parameter can be repeated and will default to -1 (discount) only if none is specified.

8.4.2 Default WebSphere Commerce commands behavior

The following commands are also part of the inventory/order/store DOM implementation. See the InfoCenter for default behavior.

- ▶ **CheckInStorePickupCmdImpl**
- ▶ **ComposeInventoryRequirementCmdImpl**

8.4.3 Security and authentication

Outbound service requests to DOM utilize WS-Security Basic Authentication with, at a minimum, transport-layer security (for example, HTTPS).

For a better understanding of how to implement web services in WebSphere Commerce, see “Understanding the WebSphere Commerce Web Service Framework” in the WebSphere Commerce InfoCenter:

<http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.webservices.doc/concepts/cwwwc55webservicesguide10.htm>

8.4.4 Database persistence

The listing of impacted WebSphere Commerce database tables used by the inventory/order/store DOM framework follows.

Store

The database tables used by the store DOM framework are:

- ▶ Physical stores:
 - STLOC
 - STLOCDS
 - STLOCATTR
- ▶ Geo nodes:
 - GEONODE
 - GEONDDS
 - GEOTREE

Inventory

Refer to 8.1, “WebSphere Commerce DOM inventory cache management” on page 196, for a discussion about configuring the following inventory caching tables:

- ▶ Cached inventory availability records: INVAVL
- ▶ Inventory configurations:
 - INVCNF
 - INVCNFREL

Order

The database tables used by the order DOM framework are:

- ▶ Physical store fulfillment centers (for order):
 - FFMCENTER
 - FFMCENTDS
 - SHPARRANGE
- ▶ Physical store addresses (for order): ADDRESS
- ▶ Physical store-fulfillment center relationships: STLFFMCREL
- ▶ Shipping charge for pickup in-store: various



A

Supporting content

The content shared in this appendix supports the implementation of the WCToMediationModule mediation module.

The WESB WCToMediationModule mediation module uses a custom stylesheet to mediate the message in OrderProcess operation. This stylesheet is named ProcessOrderToCreateOrderInput.xsl (Example A-1).

Example A-1 ProcessOrderToCreateOrderInput.xsl

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
=====
  xslt/ProcessOrderToCreateOrderInput.map
=====
-->

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xalan="http://xml.apache.org/xalan"
  xmlns:_wcf="http://www.ibm.com/xmlns/prod/commerce/9/foundation"
  xmlns:_ord="http://www.ibm.com/xmlns/prod/commerce/9/order"

  xmlns="http://www.sterlingcommerce.com/documentation/YFS/createOrder/in
  put"
```

```

exclude-result-prefixes="xalan _wcf _ord"
version="1.0">

  <xsl:output method="xml" encoding="UTF-8" omit-xml-declaration="yes"
indent="no" />
  <xsl:strip-space elements="*" />

  <xsl:template name="ProcessOrderToOrder">

    <xsl:param name="ProcessOrder" />
    <xsl:variable name="order"
select="$ProcessOrder/_ord:DataArea/_ord:Order" />

    <Order>
      <xsl:attribute name="OrderNo">
        <xsl:text>WC_</xsl:text><xsl:value-of
        select="$order/_ord:OrderIdentifier/_wcf:UniqueID" />
        </xsl:attribute>
      <xsl:attribute name="OrderDate">
        <xsl:value-of select="$order/_ord:PlacedDate" />
        </xsl:attribute>
      <OrderLines>
        <xsl:for-each select="$order/_ord:OrderItem">
          <OrderLine>
            <xsl:choose>
              <xsl:when test="contains('-',
_order:OrderItemIdentifier/_wcf:UniqueID)">
                <xsl:attribute name="PrimeLineNo">
                  <xsl:value-of
select="substring-before(_ord:OrderItemIdentifier/_wcf:UniqueID, '-')"
/>
                  </xsl:attribute>
                <xsl:attribute name="SubLineNo">
                  <xsl:value-of
select="substring-after(_ord:OrderItemIdentifier/_wcf:UniqueID, '-')"
/>
                  </xsl:attribute>
              </xsl:when>
              <xsl:otherwise>
                <xsl:attribute name="PrimeLineNo">
                  <xsl:value-of
select="_ord:OrderItemIdentifier/_wcf:UniqueID" />
                  </xsl:attribute>
              </xsl:otherwise>
            </xsl:choose>
          </OrderLine>
        </xsl:for-each>
      </OrderLines>
    </Order>
  </xsl:template>

```

```

        <xsl:attribute name="OrderedQty">
            <xsl:value-of select="_ord:Quantity" />
        </xsl:attribute>
        <xsl:attribute name="FillQuantity">
            <xsl:value-of select="_ord:Quantity" />
        </xsl:attribute>
        <xsl:attribute name="CarrierServiceCode">
            <xsl:value-of
select="_ord:OrderItemShippingInfo/_ord:ShippingMode/_ord:ShippingModeI
dentifier/_ord:ExternalIdentifier/_ord:ShipModeCode" />
            </xsl:attribute>
        <xsl:attribute name="ShipNode">
            <xsl:value-of
select="_ord:FulfillmentCenter/_ord:FulfillmentCenterIdentifier/_wcf:Na
me" />
            </xsl:attribute>
        <xsl:attribute name="ReqShipDate">
            <xsl:value-of
select="_ord:OrderItemShippingInfo/_ord:RequestedShipDate" />
            </xsl:attribute>

        <xsl:choose>
            <xsl:when

test="_ord:OrderItemShippingInfo/_ord:ShippingMode/_ord:ShippingModeIde
ntifier/_ord:ExternalIdentifier/_ord:ShipModeCode = 'PickupInStore'">
                <xsl:attribute
name="DeliveryMethod">PICK</xsl:attribute>
            </xsl:when>
            <xsl:otherwise>
                <xsl:attribute
name="DeliveryMethod">SHP</xsl:attribute>
            </xsl:otherwise>
        </xsl:choose>

        <OrderLineReservations>
            <OrderLineReservation>
                <xsl:attribute name="ItemID">
                    <xsl:value-of
select="_ord:CatalogEntryIdentifier/_wcf:ExternalIdentifier/_wcf:PartNu
mber" />
                    </xsl:attribute>
                <xsl:attribute name="Node">

```

```

        <xsl:value-of
select="_ord:FulfillmentCenter/_ord:FulfillmentCenterIdentifier/_wcf:Name" />
        </xsl:attribute>
        <xsl:attribute name="ReservationID">
        <xsl:text>WC_</xsl:text><xsl:value-of
select="$order/_ord:OrderIdentifier/_wcf:UniqueID" />
        </xsl:attribute>
        <xsl:attribute name="Quantity">
        <xsl:value-of select="_ord:Quantity" />
        </xsl:attribute>
    </OrderLineReservation>
</OrderLineReservations>

    <Item>
        <xsl:attribute name="ItemID">
        <xsl:value-of
select="_ord:CatalogEntryIdentifier/_wcf:ExternalIdentifier/_wcf:PartNumber" />
        </xsl:attribute>
        <xsl:attribute name="UnitOfMeasure">
        <xsl:choose>
            <xsl:when
test="_ord:Quantity/@uom='C62'">EACH</xsl:when>
            <xsl:when test="_ord:Quantity/@uom">
                <xsl:value-of select="_ord:Quantity/@uom"
/>
            </xsl:when>
            <xsl:otherwise>EACH</xsl:otherwise>
        </xsl:choose>
        </xsl:attribute>
    </Item>
    <PersonInfoShipTo>
        <xsl:attribute name="FirstName">
        <xsl:value-of
select="_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:ContactName/_wcf:FirstName" />
        </xsl:attribute>
        <xsl:attribute name="LastName">
        <xsl:value-of
select="_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:ContactName/_wcf:LastName" />
        </xsl:attribute>
        <xsl:attribute name="AddressLine1">

```

```

        <xsl:value-of
select="_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:Address/_w
cf:AddressLine[1]" />
        </xsl:attribute>
        <xsl:attribute name="AddressLine2">
        <xsl:value-of
select="_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:Address/_w
cf:AddressLine[2]" />
        </xsl:attribute>
        <xsl:attribute name="AddressLine3">
        <xsl:value-of
select="_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:Address/_w
cf:AddressLine[3]" />
        </xsl:attribute>
        <xsl:attribute name="City">
        <xsl:value-of
select="_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:Address/_w
cf:City" />
        </xsl:attribute>
        <xsl:attribute name="State">
        <xsl:value-of
select="_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:Address/_w
cf:StateOrProvinceName" />
        </xsl:attribute>
        <xsl:attribute name="Country">
        <xsl:value-of
select="_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:Address/_w
cf:Country" />
        </xsl:attribute>
        <xsl:attribute name="ZipCode">
        <xsl:value-of
select="_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:Address/_w
cf:PostalCode" />
        </xsl:attribute>
        <xsl:attribute name="EmailID">
        <xsl:value-of
select="_ord:OrderItemShippingInfo/_ord:ShippingAddress/_wcf:EmailAddre
ss1/_wcf:Value" />
        </xsl:attribute>
    </PersonInfoShipTo>
    <LinePriceInfo>
        <xsl:attribute name="UnitPrice">
        <xsl:value-of
select="_ord:OrderItemAmount/_wcf:UnitPrice/_wcf:Price" />
        </xsl:attribute>

```

```

        <xsl:attribute name="ListPrice">
            <xsl:value-of
select="_ord:OrderItemAmount/_wcf:UnitPrice/_wcf:Price" />
            </xsl:attribute>
            <xsl:attribute name="RetailPrice">
                <xsl:value-of
select="_ord:OrderItemAmount/_wcf:UnitPrice/_wcf:Price" />
                </xsl:attribute>
            <xsl:attribute
name="IsLinePriceForInformationOnly">N</xsl:attribute>
            <xsl:attribute
name="PricingQuantityStrategy">FIX</xsl:attribute>
            <xsl:attribute
name="IsPriceLocked">Y</xsl:attribute>
        </LinePriceInfo>
        <LineCharges>
            <xsl:variable name="amount"
select="sum(_ord:OrderItemAmount/_wcf:Adjustment/_wcf:Amount)" />
            <LineCharge>
                <xsl:attribute
name="ChargeCategory">Discount</xsl:attribute>
                <xsl:attribute name="ChargePerLine">
                    <xsl:value-of select="$amount * ($amount
&gt;= 0) - $amount * ($amount &lt; 0)" />
                </xsl:attribute>
            </LineCharge>
            <LineCharge>
                <xsl:attribute
name="ChargeCategory">Shipping</xsl:attribute>
                <xsl:attribute name="ChargeName">Shipping
Charge</xsl:attribute>
                <xsl:attribute name="ChargePerLine">
                    <xsl:value-of
select="_ord:OrderItemAmount/_wcf:ShippingCharge" />
                </xsl:attribute>
            </LineCharge>
        </LineCharges>
        <LineTaxes>
            <LineTax>
                <xsl:attribute
name="ChargeCategory">Price</xsl:attribute>
                <xsl:attribute name="Tax">
                    <xsl:value-of
select="_ord:OrderItemAmount/_wcf:SalesTax" />
                </xsl:attribute>
            </LineTax>
        </LineTaxes>
    </xsl:element>

```



```

                                <xsl:attribute name="TaxName">Sales
Tax</xsl:attribute>
                                </LineTax>
                                <LineTax>
                                    <xsl:attribute
name="ChargeCategory">Shipping</xsl:attribute>
                                    <xsl:attribute name="Tax">
                                        <xsl:value-of
select="_ord:OrderItemAmount/_wcf:ShippingTax" />
                                    </xsl:attribute>
                                    <xsl:attribute name="TaxName">Shipping
Tax</xsl:attribute>
                                </LineTax>
                                </LineTaxes>
                                </OrderLine>
                                </xsl:for-each>
                                </OrderLines>
                                <PersonInfoBillTo>
                                    <xsl:attribute name="FirstName">
                                        <xsl:value-of
select="$order/_ord:OrderPaymentInfo/_ord:PaymentInstruction/_ord:Billi
ngAddress/_wcf:ContactName/_wcf:FirstName" />
                                    </xsl:attribute>
                                    <xsl:attribute name="LastName">
                                        <xsl:value-of
select="$order/_ord:OrderPaymentInfo/_ord:PaymentInstruction/_ord:Billi
ngAddress/_wcf:ContactName/_wcf:LastName" />
                                    </xsl:attribute>
                                    <xsl:attribute name="AddressLine1">
                                        <xsl:value-of
select="$order/_ord:OrderPaymentInfo/_ord:PaymentInstruction/_ord:Billi
ngAddress/_wcf:Address/_wcf:AddressLine[1]" />
                                    </xsl:attribute>
                                    <xsl:attribute name="AddressLine2">
                                        <xsl:value-of
select="$order/_ord:OrderPaymentInfo/_ord:PaymentInstruction/_ord:Billi
ngAddress/_wcf:Address/_wcf:AddressLine[2]" />
                                    </xsl:attribute>
                                    <xsl:attribute name="AddressLine3">
                                        <xsl:value-of
select="$order/_ord:OrderPaymentInfo/_ord:PaymentInstruction/_ord:Billi
ngAddress/_wcf:Address/_wcf:AddressLine[3]" />
                                    </xsl:attribute>
                                    <xsl:attribute name="City">

```

```

        <xsl:value-of
select="$order/_ord:OrderPaymentInfo/_ord:PaymentInstruction/_ord:Billi
ngAddress/_wcf:Address/_wcf:City" />
        </xsl:attribute>
        <xsl:attribute name="State">
        <xsl:value-of
select="$order/_ord:OrderPaymentInfo/_ord:PaymentInstruction/_ord:Billi
ngAddress/_wcf:Address/_wcf:StateOrProvinceName" />
        </xsl:attribute>
        <xsl:attribute name="Country">
        <xsl:value-of
select="$order/_ord:OrderPaymentInfo/_ord:PaymentInstruction/_ord:Billi
ngAddress/_wcf:Address/_wcf:Country" />
        </xsl:attribute>
        <xsl:attribute name="ZipCode">
        <xsl:value-of
select="$order/_ord:OrderPaymentInfo/_ord:PaymentInstruction/_ord:Billi
ngAddress/_wcf:Address/_wcf:PostalCode" />
        </xsl:attribute>
        <xsl:attribute name="EMailID">
        <xsl:value-of
select="$order/_ord:OrderPaymentInfo/_ord:PaymentInstruction/_ord:Billi
ngAddress/_wcf:EmailAddress1/_wcf:Value" />
        </xsl:attribute>
        </PersonInfoBillTo>
    </Order>

</xsl:template>

</xsl:stylesheet>

```

Glossary

BOD Business object document

BOPIS Buy online, pick up in store

CEI Common Event Infrastructure

DOM Distributed Order Management

ERP Enterprise Resource Planning

HTTP Hypertext Transfer Protocol over Secure Socket Layer

HTTPS Hypertext Transfer Protocol over Secure Socket Layer

JMS Java Message Service

OAGIS Open Application Group Integration Specification

OMS Order Management System

SCA Service Component Architecture

SCA Service Component Architecture

SOA Service Oriented Architecture

SSFS Sterling Selling and Fulfillment Suite

WESB WebSphere Enterprise Service Bus

WID WebSphere Integration Developer

WMQ WebSphere MQ

WMS Warehouse Management System

WPS WebSphere Process Server

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topics in this document. Note that some publications referenced in this list might be available in softcopy only.

- ▶ *Getting Started with WebSphere Enterprise Service Bus V6*, SG24-7212
- ▶ *Mastering DynaCache in WebSphere Commerce*, SG24-7393
- ▶ *Building Multichannel Applications with WebSphere Commerce*, SG24-7787
- ▶ *DataPower Architectural Design Patterns: Integrating and Securing Services Across Domains*, SG24-7620

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at this website:

ibm.com/redbooks

Online resources

The following websites are also relevant as further information sources:

- ▶ Distributed Order Management (DOM) integration
<http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.dom-integration.doc/concepts/csmdomintro.htm>
- ▶ WebSphere Commerce tutorial on customizing within the Business Object Document (BOD) framework
http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.management-center_customization.doc/tutorial/ttf_svcmain.htm

- ▶ WebSphere Commerce Extended Sites model
<http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.admin.business.doc/concepts/cbmexten>
- ▶ Availability inventory caching
<http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.dom-integration.doc/concepts/csmdominventorycaching.htm>
- ▶ IBM WebSphere Developer Technical Journal: “Static and dynamic caching in WebSphere Application Server V5,” DeveloperWorks
http://www.ibm.com/developerworks/websphere/techjournal/0405_hines/0405_hines.html
- ▶ WebSphere Commerce Info Center V7 external application integration
<http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.integration.doc/concepts/ccvcapabilities.htm>
- ▶ WebSphere Commerce Info Center V7 Distributed Order Management Integration
<http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.domintegration.doc/concepts/csmdomintro.htm>
- ▶ WebSphere Commerce Info Center V6 Sterling Commerce reference
<http://publib.boulder.ibm.com/infocenter/wchelp/v6r0m0/topic/com.ibm.commerce.omsintegration.refapp.doc/concepts/cyaintro2.htm>
- ▶ Step-by-step guide to building a WESB mediation module for DOM integration
<http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.dom-integration.doc/tasks/tsmdombuildmedmod.htm>
- ▶ Using the DistributedMap and DistributedObjectCache interfaces for the dynamic cache
http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tdyn_distmap.html
- ▶ Object cache instance settings
http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/udyn_cacheinstancescollection.html
- ▶ Java document for class DistributedObjectCache
http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.javadoc.doc/public_html/api/com/ibm/websphere/cache/DistributedObjectCache.html

- ▶ How to configure and use WebSphere extended dynamic cache monitor
http://www.ibm.com/developerworks/websphere/downloads/cache_monitor.html
- ▶ Data service layer topic in the WebSphere Commerce Version 7 Information Center
<http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.soa.doc/concepts/csddsl.htm?resultof=%22%44%73%6c%22%20%22%64%73%6c%22%20>
- ▶ WebSphere Message Broker product site
<http://www-01.ibm.com/software/integration/wbimessagebroker/>

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



Selling and Fulfillment Solutions Using WebSphere Commerce and IBM Sterling Order Management

Sterling Commerce integration with WebSphere Commerce

Business scenarios based on latest feature pack

Sterling Commerce business solution

This IBM Redbooks publication brings together subject matter experts with experience using the leading IBM customer interaction platform for cross-channel and online commerce, IBM WebSphere Commerce, with the powerful IBM Sterling Order Management, which coordinates order fulfillment from all channels and across the extended enterprise. An integrated solution was built in the lab that illustrates how these products can be integrated to benefit IBM customers.

This publication focuses on the integration of the IBM high-volume commerce solution designed to address enterprise commerce needs by delivering a rich, robust multi-channel customer experience, with Sterling Order Management, designed to enable supplier collaboration with management and order fulfillment process optimization. By integrating WebSphere Commerce and Sterling Order Management with out-of-the-box components, we prove that customers are provided an end-to-end solution to address a complete opportunity for a fulfillment life cycle that is cost effective and easy to implement.

This publication targets a technical audience for the documentation of the integration approach by explaining the solution architecture and the implementation details. However, this publication also contains introductory chapters that contain executive summary material and provides well-documented scenarios with use cases for business analysts whose domain would be these systems.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks